

# **Program Structures and Algorithms**

## **Spring 2023(SEC 1)**

**NAME:** Harshil Shah

**NUID:** 002780887

**Assignment:** 3

**Task:** Your task for this assignment is in three parts.

(Part 1) You are to implement three (3) methods (repeat, getClock, and toMillisecs) of a class called Timer. Please see the skeleton class that I created in the repository. Timer is invoked from a class called Benchmark\_Timer which implements the Benchmark interface.

(Part 2) Implement InsertionSort (in the InsertionSort class) by simply looking up the insertion code used by Arrays.sort. If you have the instrument = true setting in test/resources/config.ini, then you will need to use the helper methods for comparing and swapping (so that they properly count the number of swaps/comparisons). The easiest is to use the helper.swapStableConditional method, continuing if it returns true, otherwise breaking the loop. Alternatively, if you are not using instrumenting, then you can write (or copy) your own compare/swap code. Either way, you must run the unit tests in InsertionSortTest.

(Part 3) Implement a main program (or you could do it via your own unit tests) to actually run the following benchmarks: measure the running times of this sort, using four different initial array ordering situations: random, ordered, partially-ordered and reverse-ordered. 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. Draw any conclusions from your observations regarding the order of growth.

### **Relationship Conclusion:**

1. For an array of random elements, the time complexity of insertion sort will be  $O(n^2)$  in the average and worst case scenario, as each element may need to be shifted multiple times before being inserted into the correct position.
2. For an array with elements in reverse order, the time complexity of insertion sort will also be  $O(n^2)$  as each element will need to be shifted to the beginning of the array for every iteration, making it the worst case scenario for insertion sort.
3. For an array which is already sorted, the time complexity of insertion sort will be  $O(n)$  as each element will only need to be checked and not shifted, making it the best case scenario for insertion sort.
4. For an array which is sorted partially, the time complexity will be somewhere between  $O(n)$  and  $O(n^2)$ , depending on the degree of sortedness of the array. The more sorted the array is, the closer the time complexity will be to  $O(n)$ .

**Evidence to support that conclusion:**

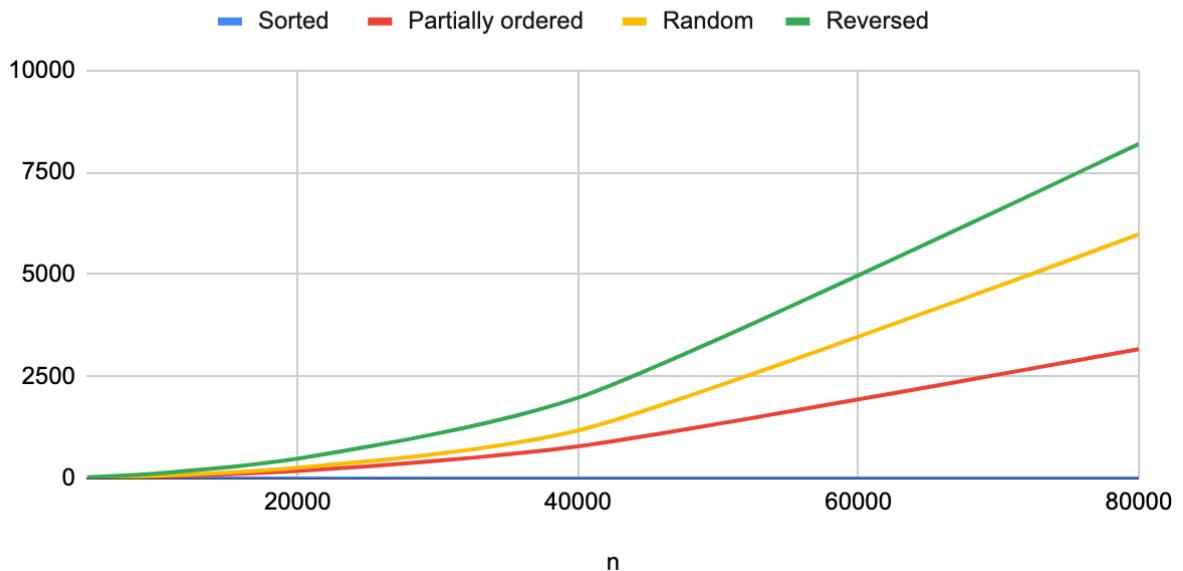
n	Sorted	Partially ordered	Random	Reversed
5000	0	11.7	15.15	30.2
10000	0	46.3	61.9	122.7
20000	0	181.45	260.6	484.4
40000	0.05	784	1172.9	1974.95
80000	0.1	3165.75	5978	8196.95

**Observation:** The graph for the time complexity of insertion sort for each of the use cases will grow as follows:

1. For an array of random elements, the graph will grow at a rate of  $O(n^2)$  which means that the growth rate of the graph will increase rapidly as the size of the array increases. The graph will form a parabolic shape.
2. For an array with elements in reverse order, the graph will grow at the same rate as for an array of random elements, which is  $O(n^2)$ . The graph will also form a parabolic shape.
3. For an array which is already sorted, the graph will grow at a rate of  $O(n)$ , which means that the growth rate of the graph will increase linearly as the size of the array increases. The graph will form a straight line.
4. For an array which is sorted partially, the graph will grow somewhere between the rate of  $O(n)$  and  $O(n^2)$ . If the array is mostly sorted, the graph will grow closer to  $O(n)$ , and if the array is less sorted, the graph will grow closer to  $O(n^2)$ . The graph will form a curve somewhere between a straight line and a parabolic shape.

## Graphical Representation:

### Comparisons - Sorted , Partially ordered , Random and Reversed



## Unit Test Screenshots:

### 1. Unit test cases for TimerTest

✓ ✓ TimerTest (edu.neu.coe.ir 2 sec 586 ms)
✓ testPauseAndLapResume0 181 ms
✓ testPauseAndLapResume1 311 ms
✓ testLap 208 ms
✓ testPause 211 ms
✓ testStop 105 ms
✓ testMillisecs 105 ms
✓ testRepeat1 121 ms
✓ testRepeat2 255 ms
✓ testRepeat3 617 ms
✓ testRepeat4 365 ms
✓ testPauseAndLap 107 ms

**TimerTest**

```

BenchmarkTest
ConfigTest
FastInverseSquareRootTest
GeoConversionsTest
LazyLoggerTest
OperationsBenchmarkTest
PrivateMethodTester
PrivateMethodTesterTest
QuickRandomTest
SorterBenchmarkTest
StatisticsTest
StatPackTest
StopwatchTest
TimerTest
BinarySearchTest

@Test
public void testPauseAndLapResume1() {
    final Timer timer = new Timer();
    GoToSleep(TENTH, which: 0);
    timer.pauseAndLap();
    GoToSleep(TENTH, which: 0);
    timer.resume();
    GoToSleep(TENTH, which: 0);
    final double time = timer.stop();
    assertEquals(TENTH_DOUBLE, time, delta: 10.0);
    assertEquals(expected: 3, run);
}

```

Run: TimerTest.testPauseAndLapResume1

Tests passed: 1 of 1 test – 391ms

TimerTest (edu.neu.coe.info621 391ms) ✓ testPauseAndLapResume1 391ms

Process finished with exit code 0

---

**util**

```

BenchmarkTest
ConfigTest
FastInverseSquareRootTest
GeoConversionsTest
LazyLoggerTest
OperationsBenchmarkTest
PrivateMethodTester
PrivateMethodTesterTest
QuickRandomTest
SorterBenchmarkTest
StatisticsTest
StatPackTest
StopwatchTest
TimerTest
BinarySearchTest

no usages  ✎ xiaohuanlin
@Test
public void testPauseAndLapResume0() {
    final Timer timer = new Timer();
    final PrivateMethodTester privateMethodTester = new PrivateMethodTester(timer);
    GoToSleep(TENTH, which: 0);
    timer.pauseAndLap();
    timer.resume();
    assertTrue(Boolean privateMethodTester.invokePrivate(name: "isRunning"));
    assertEquals(expected: 1, privateMethodTester.invokePrivate(name: "getLaps"));
}

no usages  ✎ xiaohuanlin
@Test
public void testPauseAndLapResume1() {
}

```

Run: TimerTest.testPauseAndLapResume0

Tests passed: 1 of 1 test – 175ms

TimerTest (edu.neu.coe.info621 175ms) ✓ testPauseAndLapResume0 175ms

Process finished with exit code 0

---

**union\_find**

**util**

```

ThreeSumTest
TwoSumTest
union_find
util
BenchmarkTest
ConfigTest
FastInverseSquareRootTest
GeoConversionsTest
LazyLoggerTest
OperationsBenchmarkTest
PrivateMethodTester
PrivateMethodTesterTest
QuickRandomTest
SorterBenchmarkTest
StatisticsTest
StatPackTest
StopwatchTest
TimerTest
BinarySearchTest

@Test
public void testLap() {
    final Timer timer = new Timer();
    GoToSleep(TENTH, which: 0);
    timer.lap();
    GoToSleep(TENTH, which: 0);
    final double time = timer.stop();
    assertEquals(TENTH_DOUBLE, time, delta: 10.0);
    assertEquals(expected: 2, run);
}

no usages  ✎ xiaohuanlin
@Test
public void testPause() {
    final Timer timer = new Timer();
    GoToSleep(TENTH, which: 0);
    timer.pause();
}

```

Run: TimerTest.testLap

Tests passed: 1 of 1 test – 292ms

TimerTest (edu.neu.coe.info621 292ms) ✓ testLap 292ms

Process finished with exit code 0

The image displays two screenshots of an IDE interface, likely IntelliJ IDEA, showing the execution of a Java test class named `TimerTest`.

**Top Screenshot:**

- Project Structure:** Shows packages like `ThreeSumTest`, `TwoSumTest`, `union_find`, `util`, and `BinarySearchTest`.
- Code View:** The `testPause()` method is shown, which pauses a timer for 10ms and then resumes it to check if the elapsed time is approximately 10ms.
- Run Output:** Shows the test passed in 276ms.

```
no usages  ↳ xiaohuanlin
@Test
public void testPause() {
    final Timer timer = new Timer();
    GoToSleep(TENTH, which: 0);
    timer.pause();
    GoToSleep(TENTH, which: 0);
    timer.resume();
    final double time = timer.stop();
    assertEquals(TENTH_DOUBLE, time, delta: 10.0);
    assertEquals(expected: 2, run);
}
```

**Bottom Screenshot:**

- Project Structure:** Same as the top screenshot.
- Code View:** The `testMillisecs()` method is shown, which stops a timer and then checks its millisecond value.
- Run Output:** Shows the test passed in 176ms.

```
no usages  ↳ xiaohuanlin
@Test
public void testMillisecs() {
    final Timer timer = new Timer();
    GoToSleep(TENTH, which: 0);
    timer.stop();
    final double time = timer.millisecs();
    assertEquals(TENTH_DOUBLE, time, delta: 10.0);
    assertEquals(expected: 1, run);
}

no usages  ↳ xiaohuanlin
@Test
public void testRepeat1() {
    final Timer timer = new Timer();
    final double mean = timer.repeat(n: 10, () -> {
        GoToSleep(HUNDREDTH, which: 0);
        return null;
    });
}
```

**Run Summary:** Both tests pass in approximately 176ms.

TwoSumTest

```

    > union_find
        BenchmarkTest
        ConfigTest
        FastInverseSquareRootTest
        GeoConversionsTest
        LazyLoggerTest
        OperationsBenchmarkTest
        PrivateMethodTester
        PrivateMethodTesterTest
        QuickRandomTest
        SorterBenchmarkTest
        StatisticsTest
        StatPackTest
        StopwatchTest
        TimerTest
        BinarySearchTest

```

Run: TimerTest.testRepeat1

```

@Test
public void testRepeat1() {
    final Timer timer = new Timer();
    final double mean = timer.repeat( n: 10, () -> {
        GoToSleep(HUNDRETH, which: 0);
        return null;
    });
    assertEquals( expected: 10, new PrivateMethodTester(timer).invokePrivate( name: "getLaps") );
    assertEquals( expected: TENTH_DOUBLE / 10, mean, delta: 6 );
    assertEquals( expected: 10, run );
    assertEquals( expected: 0, pre );
    assertEquals( expected: 0, post );
}

no usages ▲ xiaohuanlin
@Text

```

Tests passed: 1 of 1 test – 194 ms

TimerTest (edu.neu.coe.info621) 194 ms /Library/Java/JavaVirtualMachines/jdk-18.0.2.1.jdk/Contents/Home/bin/java ...

Process finished with exit code 0

ThreeSumTest

```

    > union_find
        BenchmarkTest
        ConfigTest
        FastInverseSquareRootTest
        GeoConversionsTest
        LazyLoggerTest
        OperationsBenchmarkTest
        PrivateMethodTester
        PrivateMethodTesterTest
        QuickRandomTest
        SorterBenchmarkTest
        StatisticsTest
        StatPackTest
        StopwatchTest
        TimerTest
        BinarySearchTest

```

Run: TimerTest.testRepeat1

```

@Text
public void testRepeat2() {
    final Timer timer = new Timer();
    final int zzz = 20;
    final double mean = timer.repeat( n: 10, () -> zzz, t -> {
        GoToSleep(t, which: 0);
        return null;
    });
    assertEquals( expected: 10, new PrivateMethodTester(timer).invokePrivate( name: "getLaps") );
    assertEquals( zzz, mean, delta: 8.5 );
    assertEquals( expected: 10, run );
    assertEquals( expected: 0, pre );
    assertEquals( expected: 0, post );
}

no usages ▲ xiaohuanlin
@Text // Slow

```

Tests passed: 1 of 1 test – 194 ms

TimerTest (edu.neu.coe.info621) 194 ms /Library/Java/JavaVirtualMachines/jdk-18.0.2.1.jdk/Contents/Home/bin/java ...

Process finished with exit code 0

symbolTable

```

    > threesum
        SourceTest
        ThreeSumTest
        TwoSumTest
    > union_find
    > util
        BenchmarkTest
        ConfigTest
        FastInverseSquareRootTest
        GeoConversionsTest
        LazyLoggerTest
        OperationsBenchmarkTest
        PrivateMethodTester
        PrivateMethodTesterTest
        QuickRandomTest
        SorterBenchmarkTest
        StatisticsTest
        StatPackTest
        StopwatchTest
        TimerTest
        BinarySearchTest

```

Run: TimerTest.testRepeat3

```

public void testRepeat3() {
    final Timer timer = new Timer();
    final int zzz = 20;
    final double mean = timer.repeat( n: 10, () -> zzz, t -> {
        GoToSleep(t, which: 0);
        return null;
    }, t -> {
        GoToSleep(t, which: -1);
        return t;
    }, t -> GoToSleep( mSecs: 10, which: 1));
    assertEquals( expected: 10, new PrivateMethodTester(timer).invokePrivate( name: "getLaps") );
    assertEquals( zzz, mean, delta: 6 );
    assertEquals( expected: 10, run );
    assertEquals( expected: 10, pre );
    assertEquals( expected: 10, post );
}

no usages ▲ xiaohuanlin
@Text // Slow

```

Tests passed: 1 of 1 test – 691 ms

TimerTest (edu.neu.coe.info621) 691 ms /Library/Java/JavaVirtualMachines/jdk-18.0.2.1.jdk/Contents/Home/bin/java ...

Process finished with exit code 0

```

144     @Test // Slow
145     public void testRepeat4() {
146         final Timer timer = new Timer();
147         final int zzz = 20;
148         final double mean = timer.repeat( n: 10,
149             () -> zzz, // supplier
150             t -> { // function
151                 result = t;
152                 GoToSleep( mSecs: 10, which: 0 );
153                 return null;
154             },
155             t -> { // pre-function
156                 GoToSleep( mSecs: 10, which: -1 );
157                 return 2*t;
158             },
159             t -> GoToSleep( mSecs: 10, which: 1 ) // post-function
160         );
161         assertEquals( expected: 10, new PrivateMethodTester(timer).invokePrivate( name: "getLaps" ) );
162         assertEquals( zzz, actual: 20, delta: 6 );
163         assertEquals( expected: 10, run );
164         assertEquals( expected: 10, pre );
165         assertEquals( expected: 10, post );
166         // This test is designed to ensure that the preFunction is properly implemented in repeat.
167         assertEquals( expected: 40, result );
168     }

```

```

27     no usages ▲ xiaohuanlin
28     @Test
29     public void testPauseAndLap() {
30         final Timer timer = new Timer();
31         final PrivateMethodTester privateMethodTester = new PrivateMethodTester(timer);
32         GoToSleep( TENTH, which: 0 );
33         timer.pauseAndLap();
34         final Long ticks = (Long) privateMethodTester.invokePrivate( name: "getTicks" );
35         assertEquals( TENTH_DOUBLE, actual: ticks / 1e6, delta: 12 );
36         assertFalse( (Boolean) privateMethodTester.invokePrivate( name: "isRunning" ) );
37         assertEquals( expected: 1, privateMethodTester.invokePrivate( name: "getLaps" ) );
38     }
39     no usages ▲ xiaohuanlin
40     @Test
41     public void testPauseAndLapResume() {
42         final Timer timer = new Timer();
43         final PrivateMethodTester privateMethodTester = new PrivateMethodTester(timer);

```

## 2. Unit tests for InsertionSort

Test Method	Execution Time
testMutatingInsertionSort	93 ms
sort0	8 ms
sort1	3 ms
sort2	3 ms
sort3	2 ms
testStaticInsertionSort	1 ms

Run: `InsertionSortTest.testMutatingInsertionSort`

```

    > reduction
      > sort
        > classic
        > counting
        > elementary
          BubbleSortTest
          InsertionSortMSDTest
          InsertionSortOptTest
          InsertionSortTest
          RandomSortTest
          SelectionSortTest
          ShellSortTest
        hashCode
        linearithmic
        BaseHelperTest
        Benchmarks
        HelperTest
        InstrumentedHelperTest
        OrderedArrayTest
        SortTest
      > symbolTable
      > threessum
        SourceTest
        ThreeSumTest
  
```

no usages `xiaohuanlin`

```

  @Test
  public void testMutatingInsertionSort() throws IOException {
    final List<Integer> list = new ArrayList<>();
    list.add(3);
    list.add(4);
    list.add(2);
    list.add(1);
    Integer[] xs = list.toArray(new Integer[0]);
    BaseHelper<Integer> helper = new BaseHelper<>(description: "InsertionSort", xs.length, Config.load(InsertionSortTest));
    GenericSort<Integer> sorter = new InsertionSort<>(helper);
    sorter.mutatingSort(xs);
    assertTrue(helper.sorted(xs));
  }
  
```

no usages `xiaohuanlin`

```

  @Test
  public void testStaticInsertionSort() throws IOException {
    final List<Integer> list = new ArrayList<>();
    list.add(3);
    list.add(4);
    list.add(2);
  }
  
```

Tests passed: 1 of 1 test - 98ms

Process finished with exit code 0

Run: `InsertionSortTest.sort0`

```

    > reduction
      > sort
        > classic
        > counting
        > elementary
          BubbleSortTest
          InsertionSortMSDTest
          InsertionSortOptTest
          InsertionSortTest
          RandomSortTest
          SelectionSortTest
          ShellSortTest
        hashCode
        linearithmic
        BaseHelperTest
        Benchmarks
        HelperTest
        InstrumentedHelperTest
        OrderedArrayTest
        SortTest
      > symbolTable
      > threessum
        SourceTest
        ThreeSumTest
  
```

`@Test`

```

public void sort0() throws Exception {
  final List<Integer> list = new ArrayList<>();
  list.add(1);
  list.add(2);
  list.add(3);
  list.add(4);
  Integer[] xs = list.toArray(new Integer[0]);
  final Config config = Config.setupConfig( instrumenting: "true", seed: "0", inversions: "1", cutoff: "", interimInversions: "0" );
  final Helper<Integer> helper = HelperFactory.create( description: "InsertionSort", list.size(), config );
  helper.init(list.size());
  final PrivateMethodTester privateMethodTester = new PrivateMethodTester(helper);
  final StatPack statPack = (StatPack) privateMethodTester.invokePrivate( name: "getStatPack" );
  SortWithHelper<Integer> sorter = new InsertionSort<>(helper);
  sorter.preProcess(xs);
  Integer[] ys = sorter.sort(xs);
  assertTrue(helper.sorted(ys));
  sorter.postProcess(ys);
  final int compares = (int) statPack.getStatistics(InstrumentedHelper.COMPARES).mean();
  assertEquals( expected: list.size() - 1, compares );
  final int inversions = (int) statPack.getStatistics(InstrumentedHelper.INVERSIONS).mean();
  assertEquals( expected: 0L, inversions );
  final int fixes = (int) statPack.getStatistics(InstrumentedHelper.FIXES).mean();
  assertEquals(inversions, fixes);
}
  
```

Tests passed: 1 of 1 test - 88ms

Process finished with exit code 0

Run: `InsertionSortTest.sort1`

```

    > reduction
      > sort
        > classic
        > counting
        > elementary
          BubbleSortTest
          InsertionSortMSDTest
          InsertionSortOptTest
          InsertionSortTest
          RandomSortTest
          SelectionSortTest
          ShellSortTest
        hashCode
        linearithmic
        BaseHelperTest
        Benchmarks
        HelperTest
        InstrumentedHelperTest
        OrderedArrayTest
        SortTest
      > symbolTable
      > threessum
        SourceTest
        ThreeSumTest
  
```

no usages `xiaohuanlin`

```

  @Test
  public void sort1() throws Exception {
    final List<Integer> list = new ArrayList<>();
    list.add(3);
    list.add(4);
    list.add(2);
    list.add(1);
    Integer[] xs = list.toArray(new Integer[0]);
    BaseHelper<Integer> helper = new BaseHelper<>( description: "InsertionSort", xs.length, Config.load(InsertionSortTest));
    GenericSort<Integer> sorter = new InsertionSort<>(helper);
    Integer[] ys = sorter.sort(xs);
    assertTrue(helper.sorted(ys));
    System.out.println(sorter.toString());
  }
  
```

no usages `xiaohuanlin`

```

  @Test
  public void testMutatingInsertionSort() throws IOException {
    final List<Integer> list = new ArrayList<>();
    list.add(3);
    list.add(4);
    list.add(2);
  }
  
```

Tests passed: 1 of 1 test - 88ms

Helper for InsertionSort with 4 elements

Process finished with exit code 0

```

    > randomwalk
    > reduction
    > sort
        > classic
        > counting
        > elementary
            BubbleSortTest
            InsertionSortMSDTest
            InsertionSortOptTest
            InsertionSortTest
            RandomSortTest
            SelectionSortTest
            ShellSortTest
        > hashCode
        > linearithmic
            BaseHelperTest
            Benchmarks
            HelperTest
            InstrumentedHelperTest
            OrderedArrayTest
            SortTest
        > symbolTable
    > threensum
        SourceTest
        ThreeSumTest
}

    77     }
    78     no usages  ± xiaohuanlin
    79     @Test
    80     public void testStaticInsertionSort() throws IOException {
    81         final List<Integer> list = new ArrayList<>();
    82         list.add(0);
    83         list.add(1);
    84         list.add(2);
    85         list.add(3);
    86         Integer[] xs = list.toArray(new Integer[0]);
    87         InsertionSort.sort(xs);
    88         assertTrue( condition: xs[0] < xs[1] && xs[1] < xs[2] && xs[2] < xs[3]);
    89     }
    90     no usages  ± xiaohuanlin+1
    91     @Test
    92     public void sort2() throws Exception {
    93         final Config config = Config.setupConfig( instrumenting: "true", seed: "0", inversions: "1", cutoff: "", interimInversions: "");
    94         int n = 100;
    95         Helper<Integer> helper = HelperFactory.create( description: "InsertionSort", n, config);
    96         helper.init(n);
    97         final PrivateMethodTester privateMethodTester = new PrivateMethodTester(helper);
}

Run:  ✓ InsertionSortTest.testStaticInsertionSort
    ✓ Tests passed: 1 of 1 test - 83ms
    ✓ InsertionSortTest (edu.neu.coe.i) 83ms /Library/Java/JavaVirtualMachines/jdk-18.0.2.1.jdk/Contents/Home/bin/java ...
        ✓ testStaticInsertionSort 83ms
    Process finished with exit code 0

    > life
    > pq
    > randomwalk
    > reduction
    > sort
        > classic
        > counting
        > elementary
            BubbleSortTest
            InsertionSortMSDTest
            InsertionSortOptTest
            InsertionSortTest
            RandomSortTest
            SelectionSortTest
            ShellSortTest
        > hashCode
        > linearithmic
            BaseHelperTest
            Benchmarks
            HelperTest
            InstrumentedHelperTest
            OrderedArrayTest
            SortTest
        > symbolTable
    > threensum
        SourceTest
        ThreeSumTest
}

    91     @Test
    92     public void sort2() throws Exception {
    93         final Config config = Config.setupConfig( instrumenting: "true", seed: "0", inversions: "1", cutoff: "", interimInversions: "");
    94         int n = 100;
    95         Helper<Integer> helper = HelperFactory.create( description: "InsertionSort", n, config);
    96         helper.init(n);
    97         final PrivateMethodTester privateMethodTester = new PrivateMethodTester(helper);
    98         final StatPack statPack = (StatPack) privateMethodTester.invokePrivate( name: "getStatPack");
    99         Integer[] xs = helper.random(Integer.class, r -> r.nextInt( bound: 1000));
   100         SortWithHelper<Integer> sorter = new InsertionSort<->(helper);
   101         sorter.preProcess(xs);
   102         Integer[] ys = sorter.sort(xs);
   103         assertEquals(helper.sorted(ys));
   104         sorter.postProcess(ys);
   105         final int compares = (int) statPack.getStatistics(InstrumentedHelper.COMPARES).mean();
   106         // NOTE: these are supposed to match within about 12%.
   107         // Since we set a specific seed, this should always succeed.
   108         // If we use true random seed and this test fails, just increase the delta a little.
   109         assertEquals( expected: 1,8 , actual: 4,8 * compares / n / (n - 1), delta: 0,12);
   110         final int inversions = (int) statPack.getStatistics(InstrumentedHelper.INVERSIONS).mean();
   111         final int fixes = (int) statPack.getStatistics(InstrumentedHelper.FIXES).mean();
   112         System.out.println(statPack);
   113         assertEquals(inversions, fixes);
   114     }
   115 }

Run:  ✓ InsertionSortTest.sort2
    ✓ Tests passed: 1 of 1 test - 80ms
    ✓ InsertionSortTest (edu.neu.coe.i) 80ms /Library/Java/JavaVirtualMachines/jdk-18.0.2.1.jdk/Contents/Home/bin/java ...
    StatPack {hits: 9,880, normalized=21.454; copies: 0, normalized=0.000; inversions: 2,421, normalized=5.257; swaps: 2,421, normalized=5.257; fixes: 2,421, normalized=5.257; compares: 2,519, normalized=5.470}
    Process finished with exit code 0

    > life
    > pq
    > randomwalk
    > reduction
    > sort
        > classic
        > counting
        > elementary
            BubbleSortTest
            InsertionSortMSDTest
            InsertionSortOptTest
            InsertionSortTest
            RandomSortTest
            SelectionSortTest
            ShellSortTest
        > hashCode
        > linearithmic
            BaseHelperTest
            Benchmarks
            HelperTest
            InstrumentedHelperTest
            OrderedArrayTest
            SortTest
        > symbolTable
    > threensum
        SourceTest
        ThreeSumTest
}

    116     @Test
    117     public void sort3() throws Exception {
    118         final Config config = Config.setupConfig( instrumenting: "true", seed: "0", inversions: "1", cutoff: "", interimInversions: "");
    119         int n = 100;
    120         Helper<Integer> helper = HelperFactory.create( description: "InsertionSort", n, config);
    121         helper.init(n);
    122         final PrivateMethodTester privateMethodTester = new PrivateMethodTester(helper);
    123         final StatPack statPack = (StatPack) privateMethodTester.invokePrivate( name: "getStatPack");
    124         Integer[] xs = new Integer[n];
    125         for (int i = 0; i < n; i++) xs[i] = n - i;
    126         SortWithHelper<Integer> sorter = new InsertionSort<->(helper);
    127         sorter.preProcess(xs);
    128         Integer[] ys = sorter.sort(xs);
    129         assertEquals(helper.sorted(ys));
    130         sorter.postProcess(ys);
    131         final int compares = (int) statPack.getStatistics(InstrumentedHelper.COMPARES).mean();
    132         // NOTE: these are supposed to match within about 12%.
    133         // Since we set a specific seed, this should always succeed.
    134         // If we use true random seed and this test fails, just increase the delta a little.
    135         assertEquals( expected: 4950, compares);
    136         final int inversions = (int) statPack.getStatistics(InstrumentedHelper.INVERSIONS).mean();
    137         final int fixes = (int) statPack.getStatistics(InstrumentedHelper.FIXES).mean();
    138         System.out.println(statPack);
    139         assertEquals(inversions, fixes);
    140     }
    141 }

Run:  ✓ InsertionSortTest.sort3
    ✓ Tests passed: 1 of 1 test - 83ms
    ✓ InsertionSortTest (edu.neu.coe.i) 83ms /Library/Java/JavaVirtualMachines/jdk-18.0.2.1.jdk/Contents/Home/bin/java ...
    StatPack {hits: 19,800, normalized=42.995; copies: 0, normalized=0.000; inversions: 4,958, normalized=10.749; swaps: 4,958, normalized=10.749; fixes: 4,958, normalized=10.749; compares: 4,958, normalized=10.749}
    Process finished with exit code 0

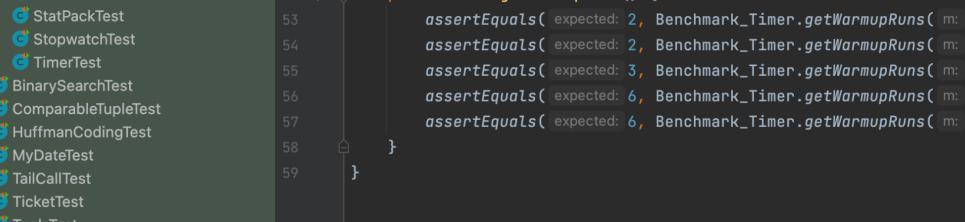
```

### 3. Unit tests for BenchmarkTest

The screenshot shows a Java IDE interface with three main sections:

- Top Panel:** Shows the results of a **BenchmarkTest**. It lists two tests: **testWaitPeriods** (1sec 405 ms) and **getWarmupRuns** (0 ms). Both tests are marked with a green checkmark.
- Middle Panel:** Displays the source code for the **BenchmarkTest** class. The **testWaitPeriods** method is highlighted. The code uses the **Benchmark** annotation to define three execution phases with different sleep durations (100ms, 200ms, and 50ms). It also includes assertions to verify the number of runs and warmups.
- Bottom Panel:** Shows the output of the **BenchmarkTest.testWaitPeriods** run. It displays the command used (BenchmarkTest edu.neu.1sec 429 ms), the log message "Begin run: testWaitPeriods with 2 runs", and the exit status "Process finished with exit code 0".

```
public void testWaitPeriods() throws Exception {
    int nRuns = 2;
    int warmups = 2;
    Benchmark<Boolean> bm = new Benchmark_Timer<>(
        description: "testWaitPeriods", b -> {
            GoToSleep( mSecs: 100L, which: -1);
            return null;
        },
        b -> {
            GoToSleep( mSecs: 200L, which: 0);
        },
        b -> {
            GoToSleep( mSecs: 50L, which: 1);
        });
    double x = bm.run( t: true, nRuns);
    assertEquals(nRuns, post);
    assertEquals( expected: nRuns + warmups, run);
    assertEquals( expected: nRuns + warmups, pre);
    assertEquals( expected: 200, x, delta: 10);
}
```



```
PrivateMethodTesterTest
QuickRandomTest
SorterBenchmarkTest
StatisticsTest
StatPackTest
StopwatchTest
TimerTest
BinarySearchTest
ComparableTupleTest
HuffmanCodingTest
MyDateTest
TailCallTest
TicketTest
TupleTest

> resources
target
.gitignore
n: BenchmarkTest.getWarmupRuns x
  ✓ Tests passed: 1 of 1 test - 67 ms
  ✓ BenchmarkTest (edu.neu.coe.inl) 67 ms
    ✓ getWarmupRuns 67 ms
/Library/Java/JavaVirtualMachines/jdk-18.0.2.1.jdk/Contents/Home/bin/java ...
Process finished with exit code 0
```