INFO6205 Assignment 3 (Benchmark)

NAME: Bohan Feng

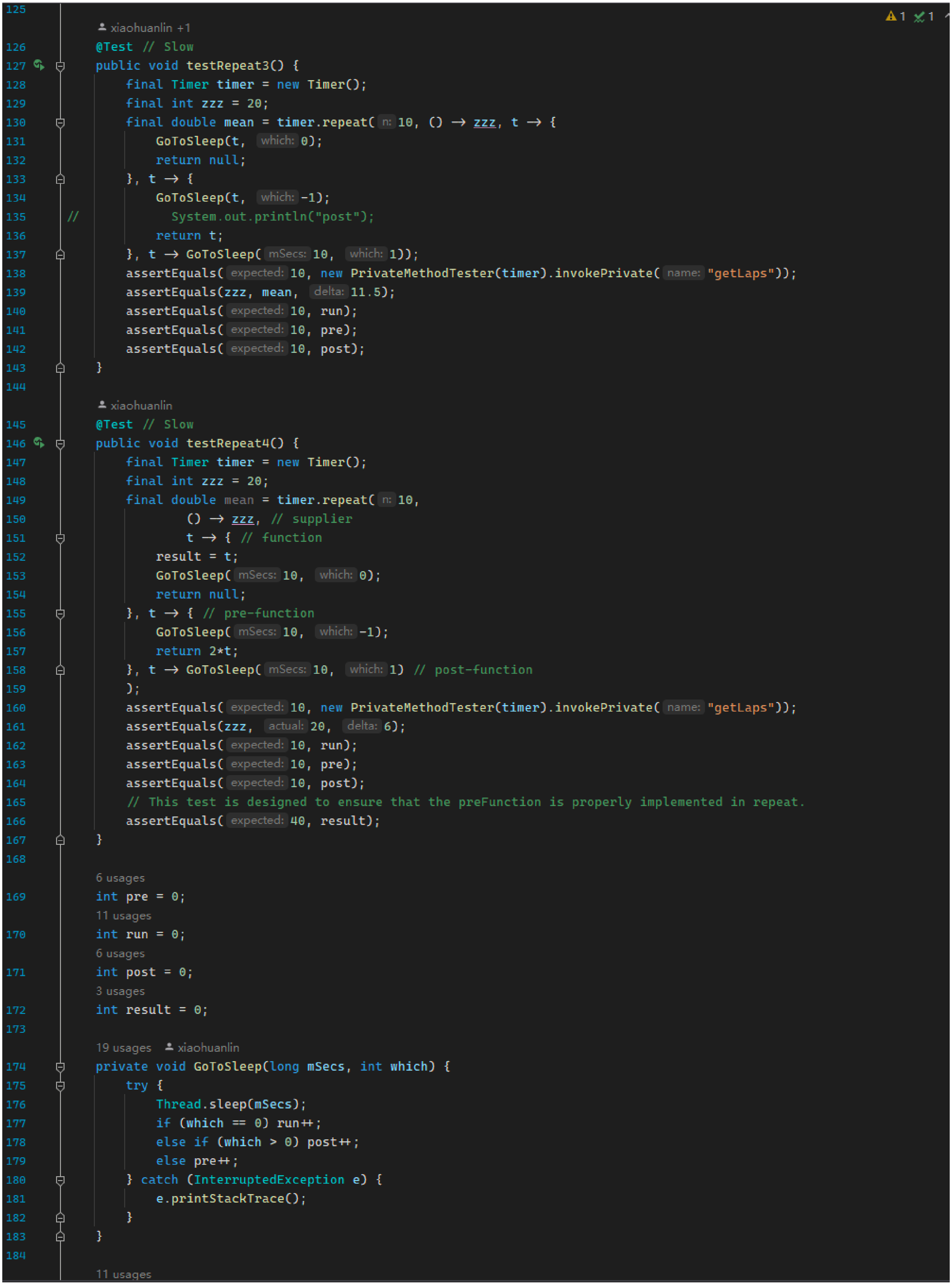
NUID: 001564249

Repository: https://github.com/fengb3/INFO6205

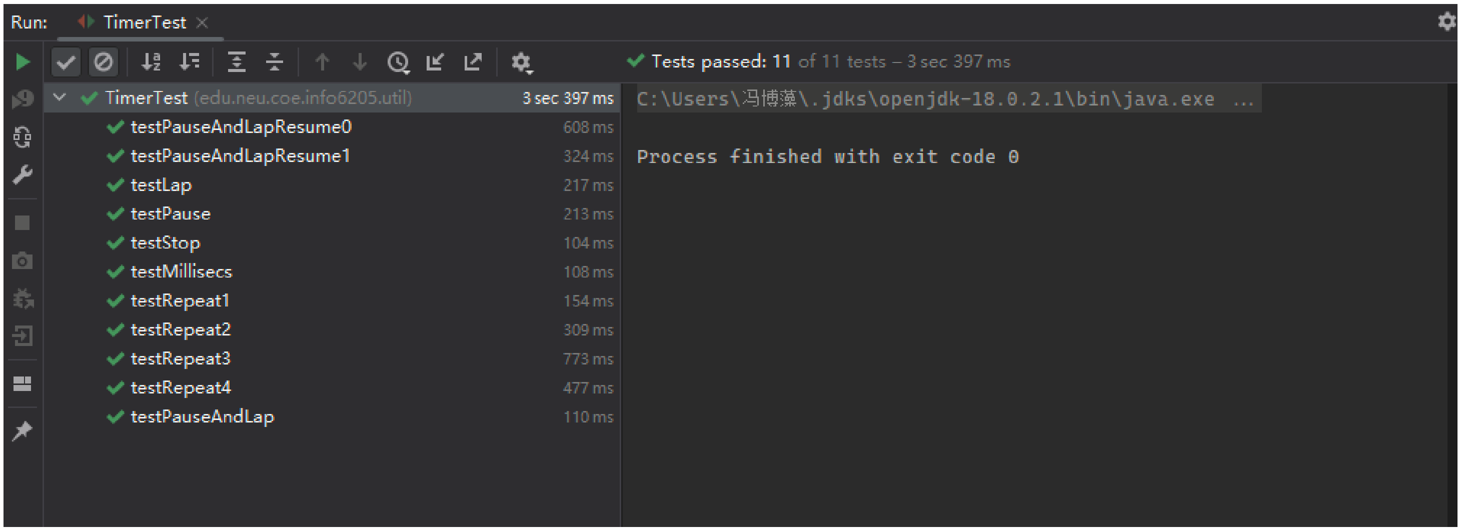
Part 1 Timer & Benchmark

Unit Test Screenshot

Timer:

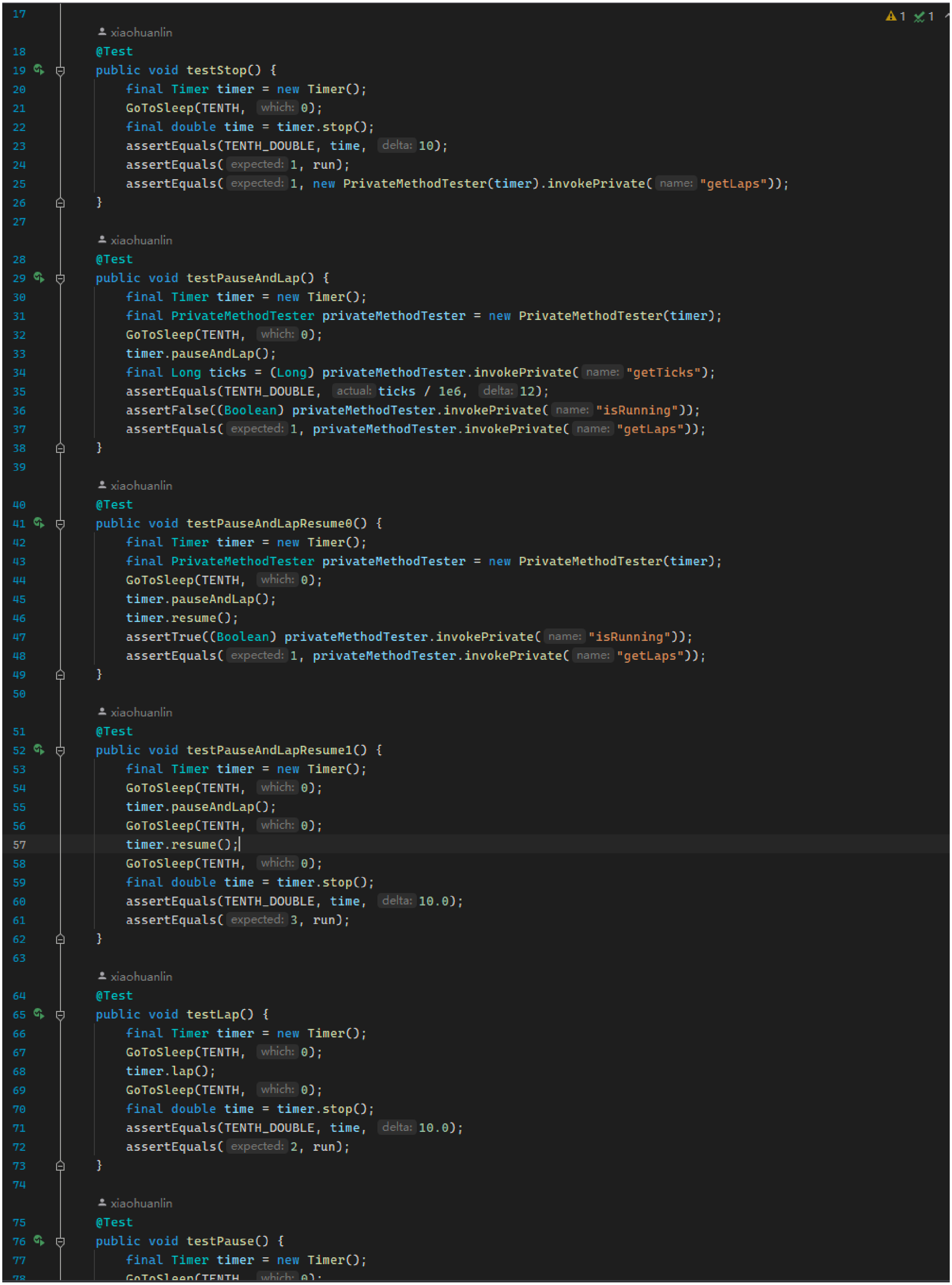


```
    xiaohuanlin +1
    @Test // Slow
    public void testRepeat3() {
        final Timer timer = new Timer();
        final int zzz = 20;
        final double mean = timer.repeat( n: 10, () -> zzz, t -> {
            GoToSleep(t,  which: 0);
            return null;
        }, t -> {
            GoToSleep(t,  which: -1);
//            System.out.println("post");
            return t;
        }, t -> GoToSleep( mSecs: 10,  which: 1));
        assertEquals( expected: 10, new PrivateMethodTester(timer).invokePrivate( name: "getLaps"));
        assertEquals(zzz, mean,  delta: 11.5);
        assertEquals( expected: 10, run);
        assertEquals( expected: 10, pre);
        assertEquals( expected: 10, post);
    }

    xiaohuanlin
    @Test // Slow
    public void testRepeat4() {
        final Timer timer = new Timer();
        final int zzz = 20;
        final double mean = timer.repeat( n: 10,
                () -> zzz, // supplier
                t -> { // function
            result = t;
            GoToSleep( mSecs: 10,  which: 0);
            return null;
        }, t -> { // pre-function
            GoToSleep( mSecs: 10,  which: -1);
            return 2*t;
        }, t -> GoToSleep( mSecs: 10,  which: 1) // post-function
        );
        assertEquals( expected: 10, new PrivateMethodTester(timer).invokePrivate( name: "getLaps"));
        assertEquals(zzz,  actual: 20,  delta: 6);
        assertEquals( expected: 10, run);
        assertEquals( expected: 10, pre);
        assertEquals( expected: 10, post);
        // This test is designed to ensure that the preFunction is properly implemented in repeat.
        assertEquals( expected: 40, result);
    }

    6 usages
    int pre = 0;
    11 usages
    int run = 0;
    6 usages
    int post = 0;
    3 usages
    int result = 0;

    19 usages  xiaohuanlin
    private void GoToSleep(long mSecs, int which) {
        try {
            Thread.sleep(mSecs);
            if (which == 0) run++;
            else if (which > 0) post++;
            else pre++;
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }

    11 usages
```

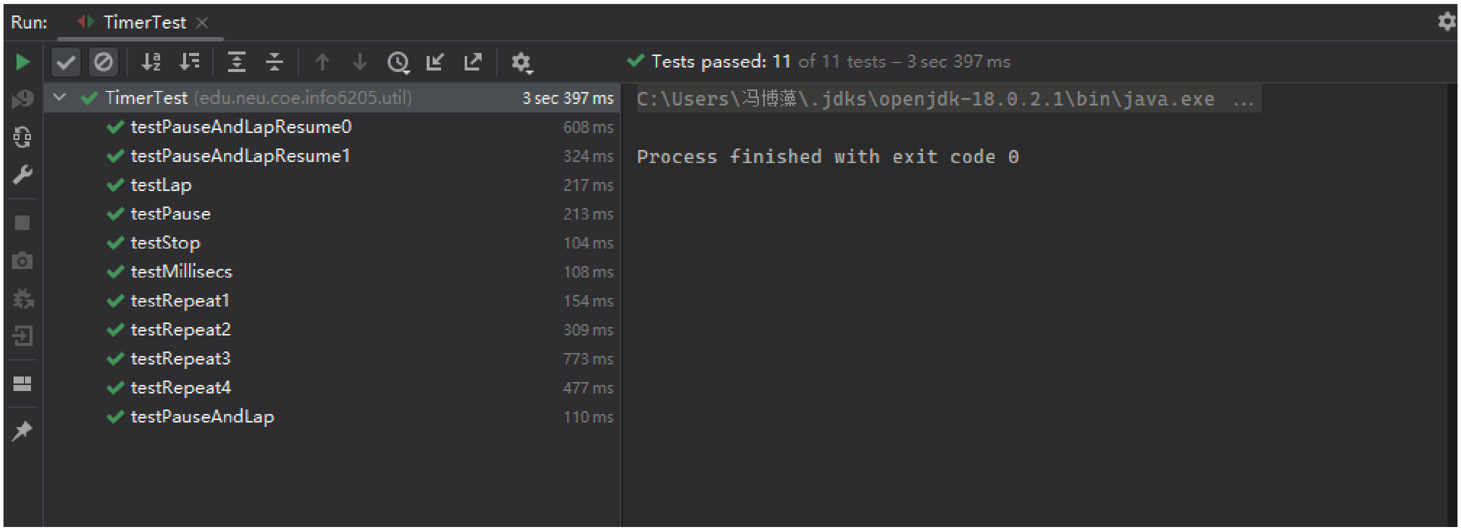


```
✓ Tests passed: 11 of 11 tests – 3 sec 397 ms

✓ TimerTest (edu.neu.coe.info6205.util)   3 sec 397 ms
    ✓ testPauseAndLapResume0               608 ms
    ✓ testPauseAndLapResume1               324 ms
    ✓ testLap                              217 ms
    ✓ testPause                            213 ms
    ✓ testStop                             104 ms
    ✓ testMillisecs                        108 ms
    ✓ testRepeat1                          154 ms
    ✓ testRepeat2                          309 ms
    ✓ testRepeat3                          773 ms
    ✓ testRepeat4                          477 ms
    ✓ testPauseAndLap                      110 ms
```

```
C:\Users\冯博藻\.jdks\openjdk-18.0.2.1\bin\java.exe  ...

Process finished with exit code 0
```



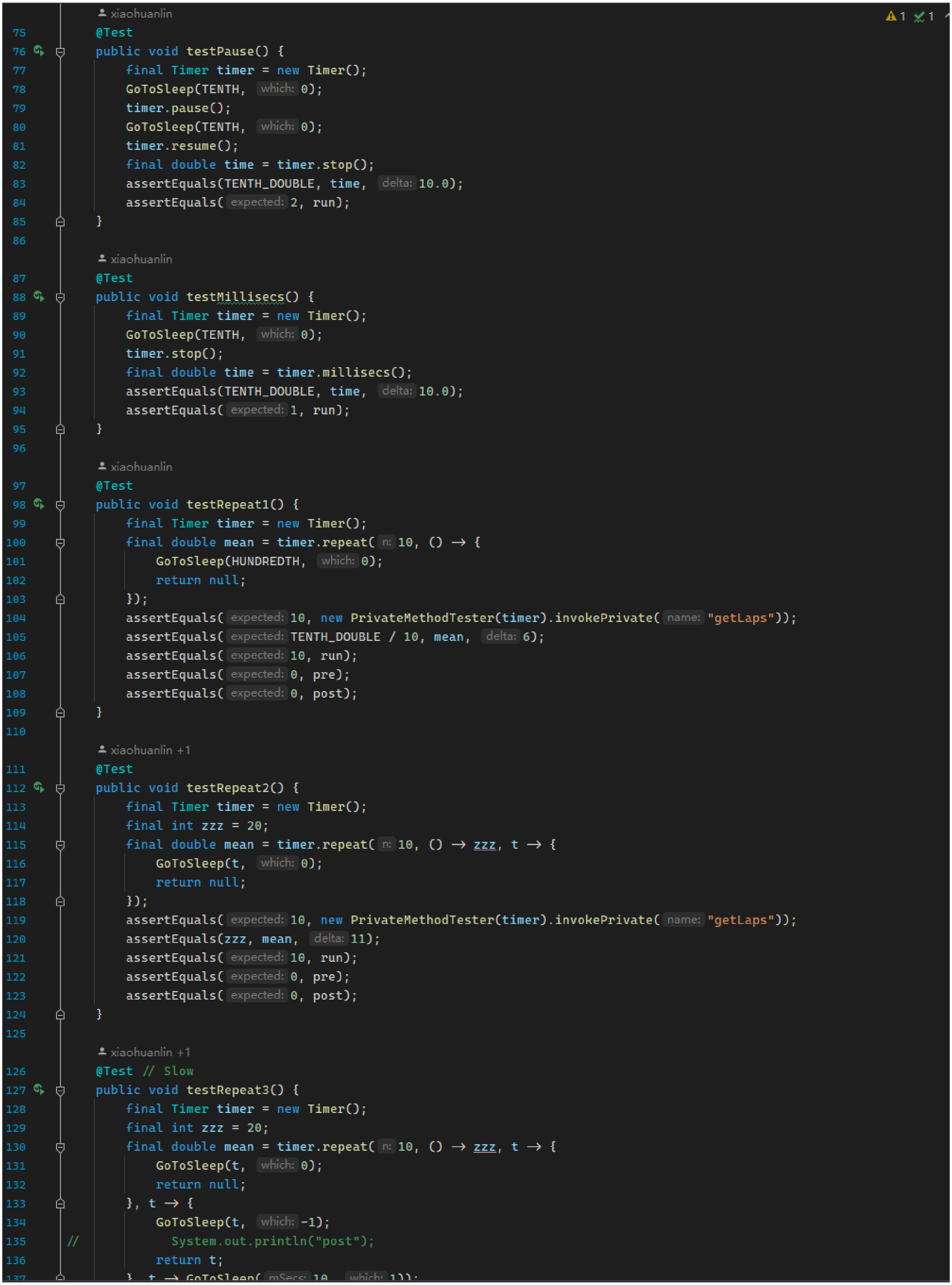
```
    xiaohuanlin
    @Test
    public void testStop() {
        final Timer timer = new Timer();
        GoToSleep(TENTH,  which: 0);
        final double time = timer.stop();
        assertEquals(TENTH_DOUBLE, time,  delta: 10);
        assertEquals( expected: 1, run);
        assertEquals( expected: 1, new PrivateMethodTester(timer).invokePrivate( name: "getLaps"));
    }

    xiaohuanlin
    @Test
    public void testPauseAndLap() {
        final Timer timer = new Timer();
        final PrivateMethodTester privateMethodTester = new PrivateMethodTester(timer);
        GoToSleep(TENTH,  which: 0);
        timer.pauseAndLap();
        final Long ticks = (Long) privateMethodTester.invokePrivate( name: "getTicks");
        assertEquals(TENTH_DOUBLE,  actual: ticks / 1e6,  delta: 12);
        assertFalse((Boolean) privateMethodTester.invokePrivate( name: "isRunning"));
        assertEquals( expected: 1, privateMethodTester.invokePrivate( name: "getLaps"));
    }

    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"));
    }

    xiaohuanlin
    @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);
    }

    xiaohuanlin
    @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);
    }

    xiaohuanlin
    @Test
    public void testPause() {
        final Timer timer = new Timer();
        GoToSleep(TENTH,  which: 0);
```



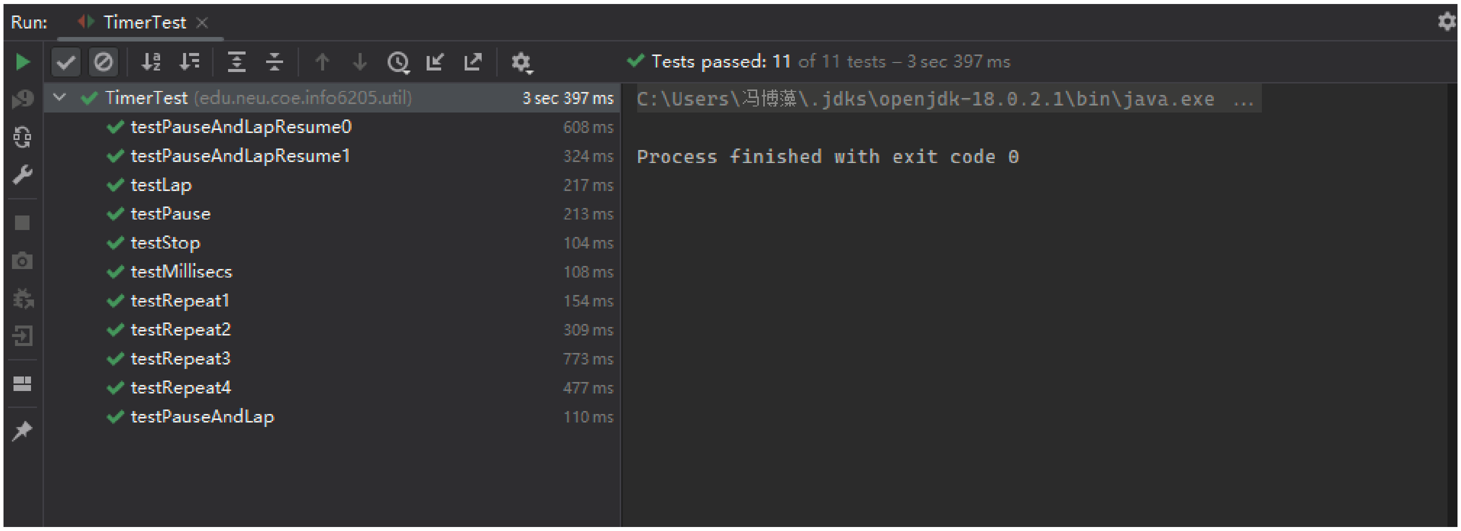
✓ Tests passed: 11 of 11 tests – 3 sec 397 ms

```
✓ TimerTest (edu.neu.coe.info6205.util)      3 sec 397 ms
    ✓ testPauseAndLapResume0                       608 ms
    ✓ testPauseAndLapResume1                       324 ms
    ✓ testLap                                      217 ms
    ✓ testPause                                    213 ms
    ✓ testStop                                     104 ms
    ✓ testMillisecs                                108 ms
    ✓ testRepeat1                                  154 ms
    ✓ testRepeat2                                  309 ms
    ✓ testRepeat3                                  773 ms
    ✓ testRepeat4                                  477 ms
    ✓ testPauseAndLap                              110 ms
```

```
C:\Users\冯博藻\.jdks\openjdk-18.0.2.1\bin\java.exe ...

Process finished with exit code 0
```



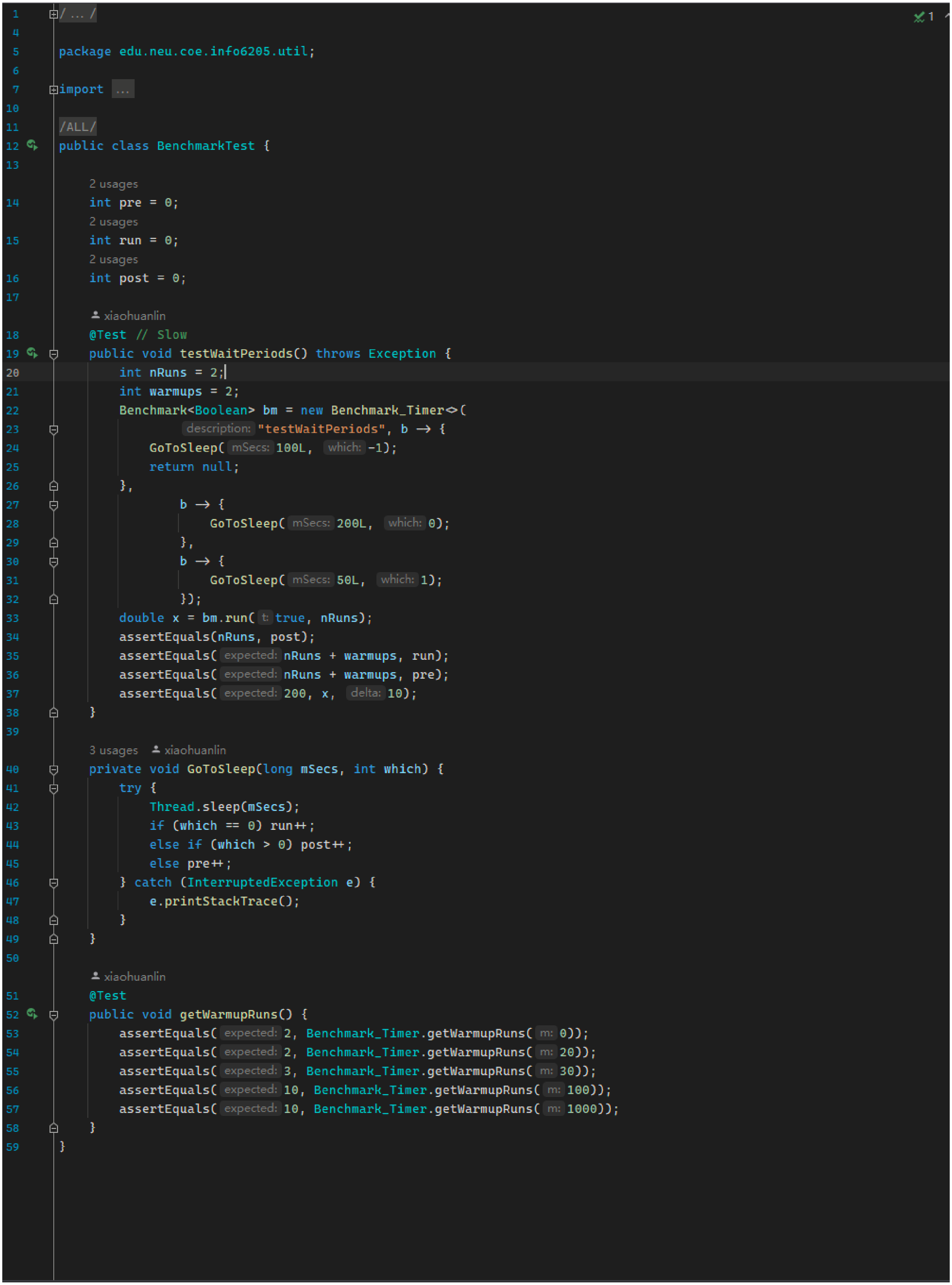
```
    xiaohuanlin
    @Test
    public void testPause() {
        final Timer timer = new Timer();
        GoToSleep(TENTH,  which: 0);
        timer.pause();
        GoToSleep(TENTH,  which: 0);
        timer.resume();
        final double time = timer.stop();
        assertEquals(TENTH_DOUBLE, time,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

    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);
    }

    xiaohuanlin
    @Test
    public void testRepeat1() {
        final Timer timer = new Timer();
        final double mean = timer.repeat( n: 10, () → {
            GoToSleep(HUNDREDTH,  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);
    }

    xiaohuanlin +1
    @Test
    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: 11);
        assertEquals( expected: 10, run);
        assertEquals( expected: 0, pre);
        assertEquals( expected: 0, post);
    }

    xiaohuanlin +1
    @Test // Slow
    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);
//            System.out.println("post");
            return t;
        }, t → GoToSleep( mSecs: 10,  which: 1));
```



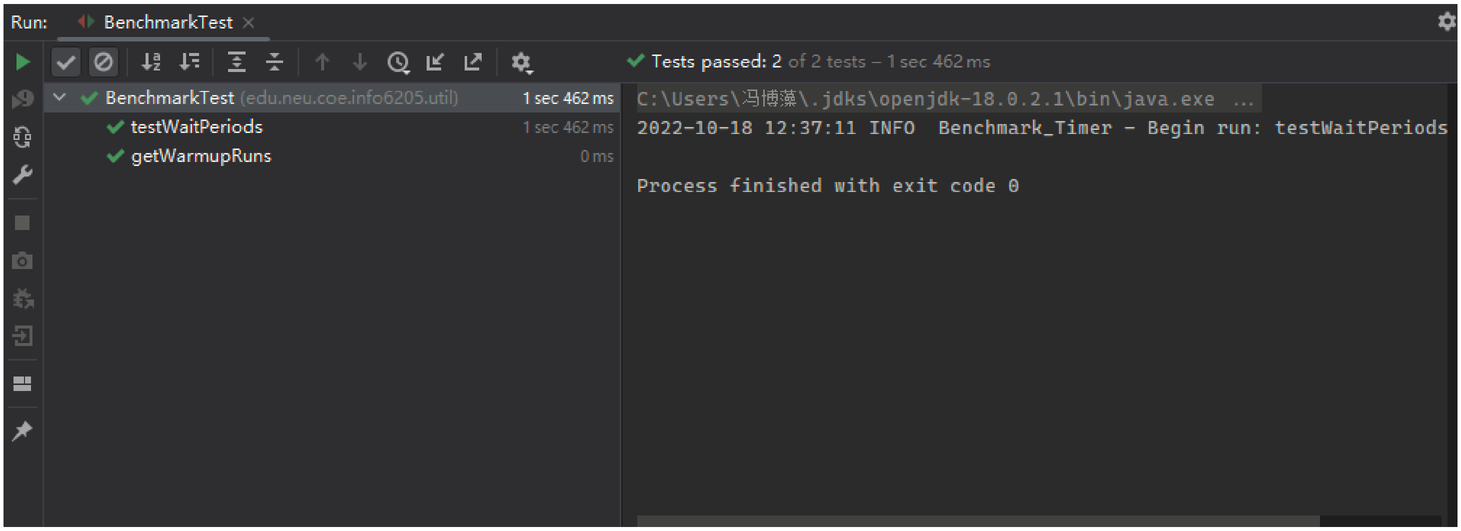
```
✓ Tests passed: 11 of 11 tests – 3 sec 397 ms

✓ TimerTest (edu.neu.coe.info6205.util)   3 sec 397 ms
    ✓ testPauseAndLapResume0                608 ms
    ✓ testPauseAndLapResume1                324 ms
    ✓ testLap                               217 ms
    ✓ testPause                             213 ms
    ✓ testStop                              104 ms
    ✓ testMillisecs                         108 ms
    ✓ testRepeat1                           154 ms
    ✓ testRepeat2                           309 ms
    ✓ testRepeat3                           773 ms
    ✓ testRepeat4                           477 ms
    ✓ testPauseAndLap                       110 ms
```

```
C:\Users\冯博藻\.jdks\openjdk-18.0.2.1\bin\java.exe ...

Process finished with exit code 0
```

Benchmark:



```
/ ... /

package edu.neu.coe.info6205.util;

import ...

/ALL/
public class BenchmarkTest {

    2 usages
    int pre = 0;
    2 usages
    int run = 0;
    2 usages
    int post = 0;

    xiaohuanlin
    @Test // Slow
    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);
    }

    3 usages   xiaohuanlin
    private void GoToSleep(long mSecs, int which) {
        try {
            Thread.sleep(mSecs);
            if (which == 0) run++;
            else if (which > 0) post++;
            else pre++;
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }

    xiaohuanlin
    @Test
    public void getWarmupRuns() {
        assertEquals( expected: 2, Benchmark_Timer.getWarmupRuns( m: 0));
        assertEquals( expected: 2, Benchmark_Timer.getWarmupRuns( m: 20));
        assertEquals( expected: 3, Benchmark_Timer.getWarmupRuns( m: 30));
        assertEquals( expected: 10, Benchmark_Timer.getWarmupRuns( m: 100));
        assertEquals( expected: 10, Benchmark_Timer.getWarmupRuns( m: 1000));
    }
}
```



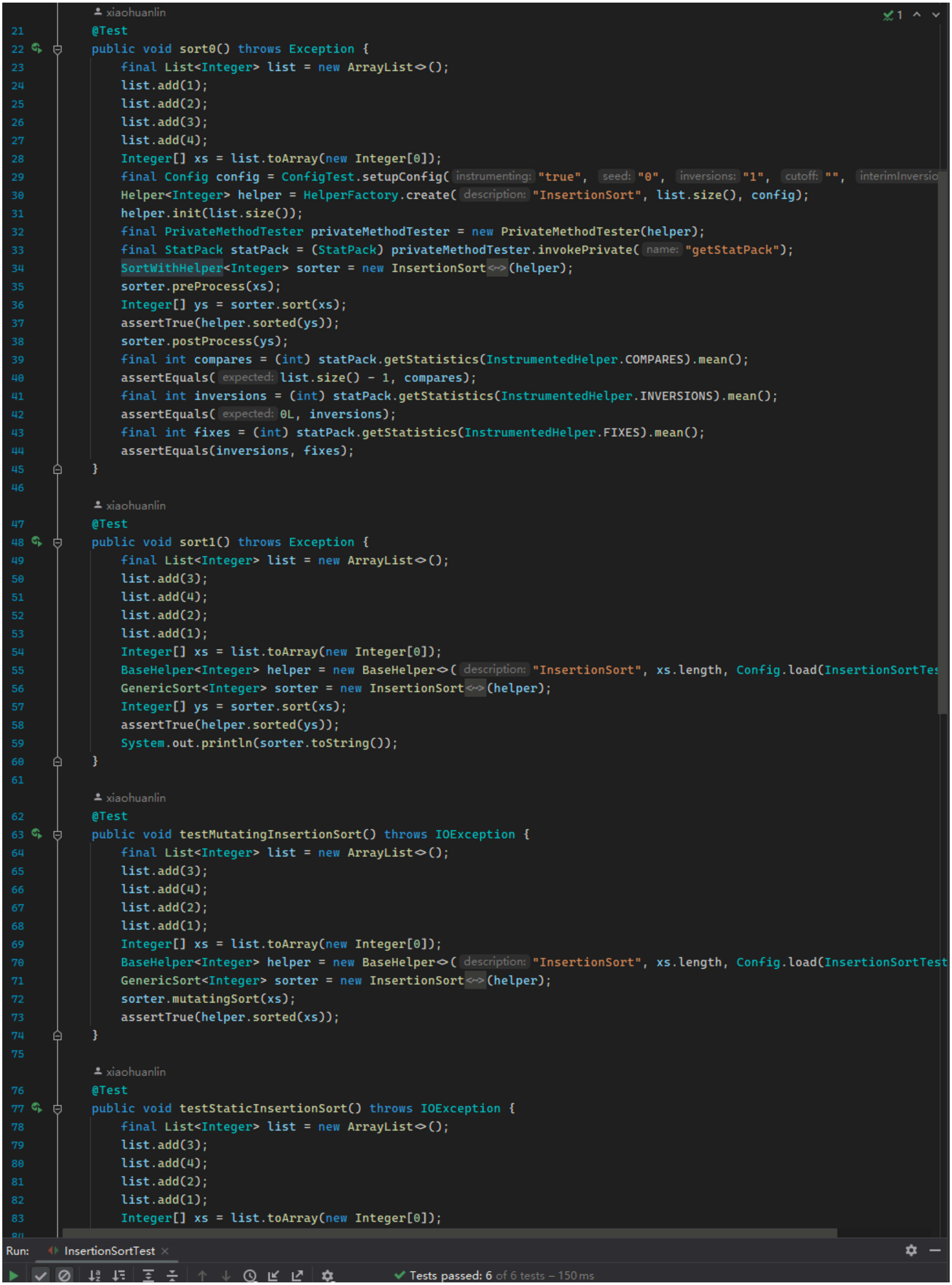
✓ Tests passed: 2 of 2 tests – 1 sec 462 ms

✓ BenchmarkTest (edu.neu.coe.info6205.util) 1 sec 462 ms
✓ testWaitPeriods 1 sec 462 ms
✓ getWarmupRuns 0 ms

```
C:\Users\冯博藻\.jdks\openjdk-18.0.2.1\bin\java.exe ...
2022-10-18 12:37:11 INFO  Benchmark_Timer - Begin run: testWaitPeriods

Process finished with exit code 0
```

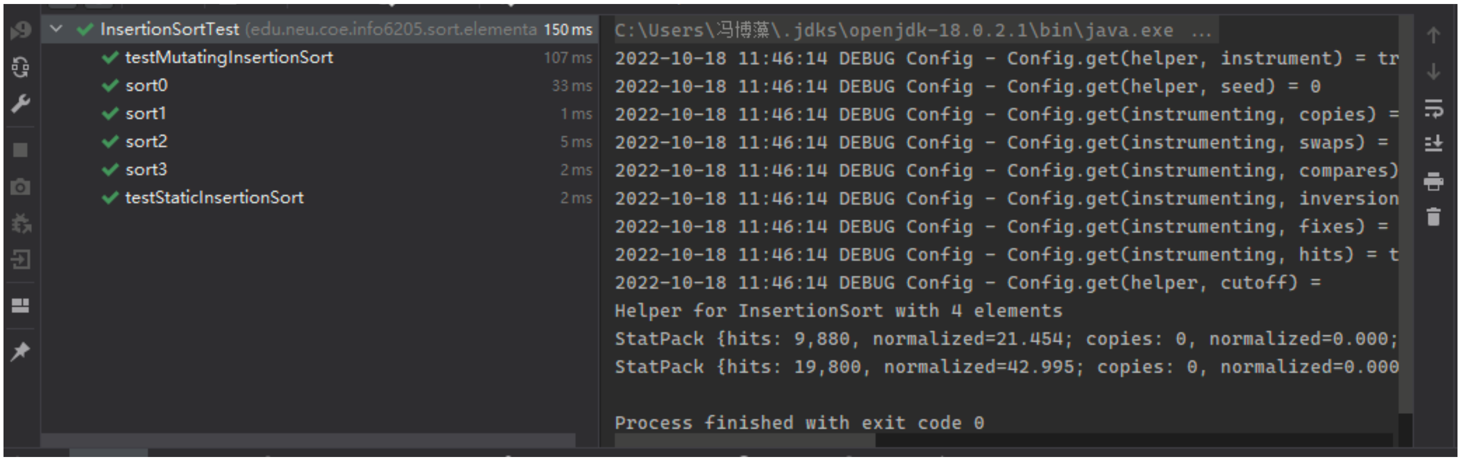
Part 2 Inserrtion Sort

Unit Test Screetshot:



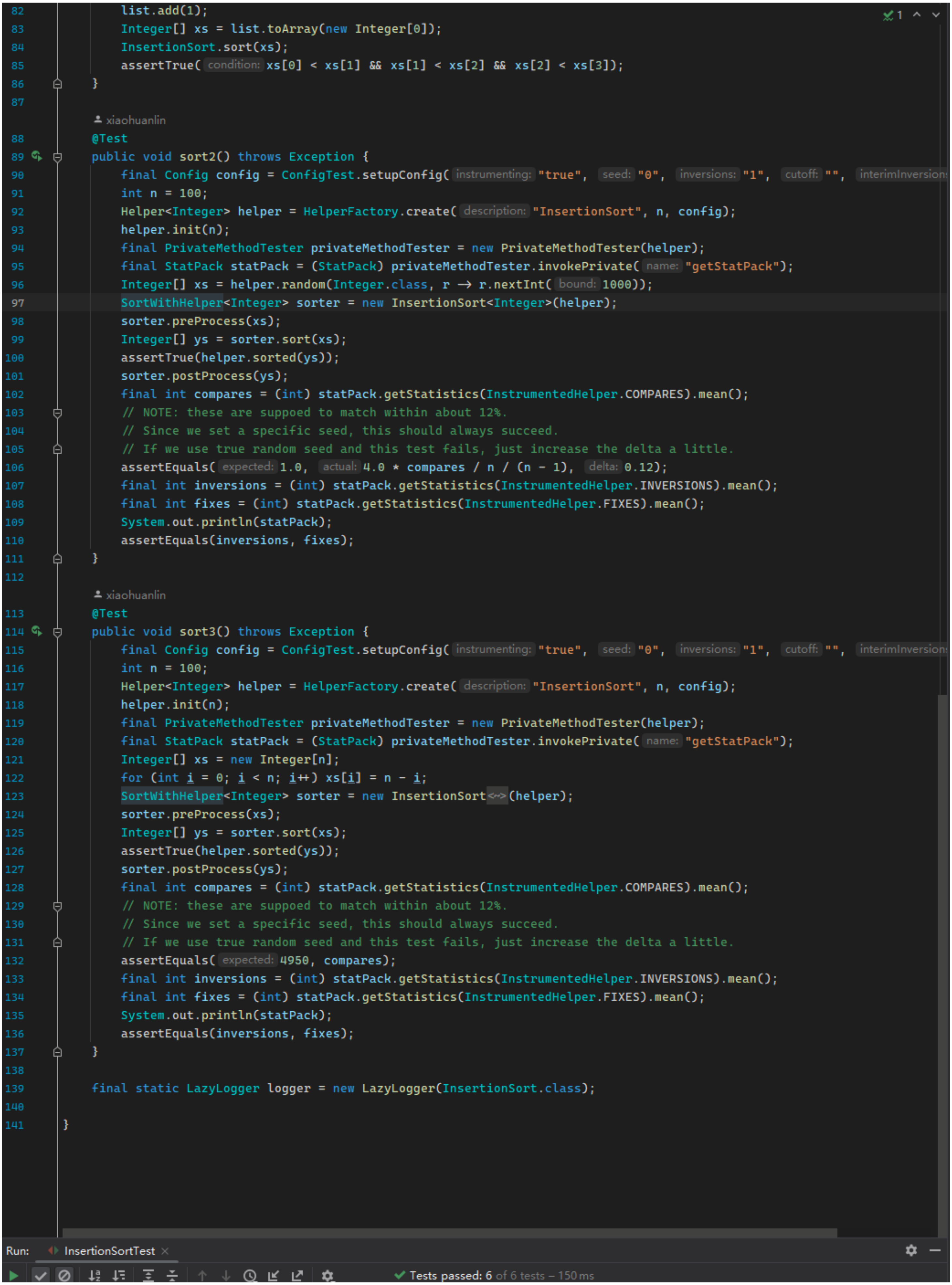
```
    xiaohuanlin
    @Test
    public void sort0() throws Exception {
        final List<Integer> list = new ArrayList<>();
        list.add(1);
        list.add(2);
        list.add(3);
        list.add(4);
        Integer[] xs = list.toArray(new Integer[0]);
        final Config config = ConfigTest.setupConfig( instrumenting: "true",  seed: "0",  inversions: "1",  cutoff: "",  interimInversio
        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);
    }

    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(InsertionSortTes
        GenericSort<Integer> sorter = new InsertionSort<~>(helper);
        Integer[] ys = sorter.sort(xs);
        assertTrue(helper.sorted(ys));
        System.out.println(sorter.toString());
    }

    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));
    }

    xiaohuanlin
    @Test
    public void testStaticInsertionSort() 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]);
```

✓ Tests passed: 6 of 6 tests – 150 ms



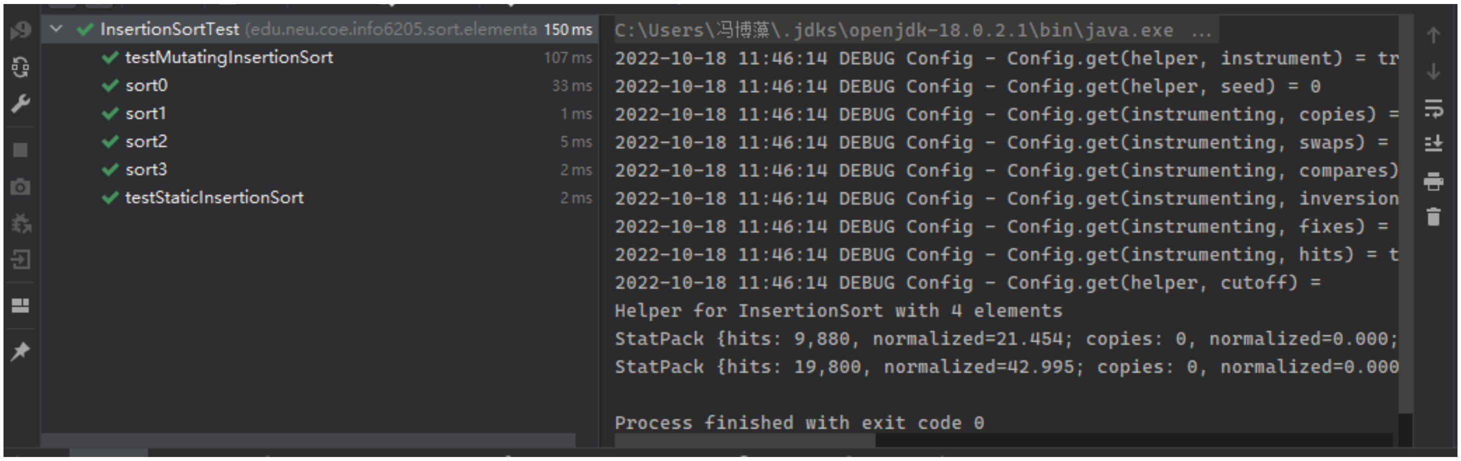
```
InsertionSortTest (edu.neu.coe.info6205.sort.elementa 150 ms
    testMutatingInsertionSort                         107 ms
    sort0                                              33 ms
    sort1                                               1 ms
    sort2                                               5 ms
    sort3                                               2 ms
    testStaticInsertionSort                             2 ms
```

```
C:\Users\冯博藻\.jdks\openjdk-18.0.2.1\bin\java.exe ...
2022-10-18 11:46:14 DEBUG Config - Config.get(helper, instrument) = tr
2022-10-18 11:46:14 DEBUG Config - Config.get(helper, seed) = 0
2022-10-18 11:46:14 DEBUG Config - Config.get(instrumenting, copies) =
2022-10-18 11:46:14 DEBUG Config - Config.get(instrumenting, swaps) =
2022-10-18 11:46:14 DEBUG Config - Config.get(instrumenting, compares)
2022-10-18 11:46:14 DEBUG Config - Config.get(instrumenting, inversion
2022-10-18 11:46:14 DEBUG Config - Config.get(instrumenting, fixes) =
2022-10-18 11:46:14 DEBUG Config - Config.get(instrumenting, hits) = t
2022-10-18 11:46:14 DEBUG Config - Config.get(helper, cutoff) =
Helper for InsertionSort with 4 elements
StatPack {hits: 9,880, normalized=21.454; copies: 0, normalized=0.000;
StatPack {hits: 19,800, normalized=42.995; copies: 0, normalized=0.000

Process finished with exit code 0
```



```
        list.add(1);
        Integer[] xs = list.toArray(new Integer[0]);
        InsertionSort.sort(xs);
        assertTrue( condition: xs[0] < xs[1] && xs[1] < xs[2] && xs[2] < xs[3]);
    }

    xiaohuanlin
    @Test
    public void sort2() throws Exception {
        final Config config = ConfigTest.setupConfig( instrumenting: "true",  seed: "0",  inversions: "1",  cutoff: "",  interimInversion
        int n = 100;
        Helper<Integer> helper = HelperFactory.create( description: "InsertionSort", n, config);
        helper.init(n);
        final PrivateMethodTester privateMethodTester = new PrivateMethodTester(helper);
        final StatPack statPack = (StatPack) privateMethodTester.invokePrivate( name: "getStatPack");
        Integer[] xs = helper.random(Integer.class, r -> r.nextInt( bound: 1000));
        SortWithHelper<Integer> sorter = new InsertionSort<Integer>(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();
        // NOTE: these are supposed to match within about 12%.
        // Since we set a specific seed, this should always succeed.
        // If we use true random seed and this test fails, just increase the delta a little.
        assertEquals( expected: 1.0,  actual: 4.0 * compares / n / (n - 1),  delta: 0.12);
        final int inversions = (int) statPack.getStatistics(InstrumentedHelper.INVERSIONS).mean();
        final int fixes = (int) statPack.getStatistics(InstrumentedHelper.FIXES).mean();
        System.out.println(statPack);
        assertEquals(inversions, fixes);
    }

    xiaohuanlin
    @Test
    public void sort3() throws Exception {
        final Config config = ConfigTest.setupConfig( instrumenting: "true",  seed: "0",  inversions: "1",  cutoff: "",  interimInversion
        int n = 100;
        Helper<Integer> helper = HelperFactory.create( description: "InsertionSort", n, config);
        helper.init(n);
        final PrivateMethodTester privateMethodTester = new PrivateMethodTester(helper);
        final StatPack statPack = (StatPack) privateMethodTester.invokePrivate( name: "getStatPack");
        Integer[] xs = new Integer[n];
        for (int i = 0; i < n; i++) xs[i] = n - i;
        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();
        // NOTE: these are supposed to match within about 12%.
        // Since we set a specific seed, this should always succeed.
        // If we use true random seed and this test fails, just increase the delta a little.
        assertEquals( expected: 4950, compares);
        final int inversions = (int) statPack.getStatistics(InstrumentedHelper.INVERSIONS).mean();
        final int fixes = (int) statPack.getStatistics(InstrumentedHelper.FIXES).mean();
        System.out.println(statPack);
        assertEquals(inversions, fixes);
    }

    final static LazyLogger logger = new LazyLogger(InsertionSort.class);

}
```

✓ Tests passed: 6 of 6 tests – 150 ms



```
InsertionSortTest (edu.neu.coe.info6205.sort.elementa 150 ms
    testMutatingInsertionSort                        107 ms
    sort0                                             33 ms
    sort1                                              1 ms
    sort2                                              5 ms
    sort3                                              2 ms
    testStaticInsertionSort                            2 ms
```

```
C:\Users\冯博藻\.jdks\openjdk-18.0.2.1\bin\java.exe ...
2022-10-18 11:46:14 DEBUG Config - Config.get(helper, instrument) = tr
2022-10-18 11:46:14 DEBUG Config - Config.get(helper, seed) = 0
2022-10-18 11:46:14 DEBUG Config - Config.get(instrumenting, copies) =
2022-10-18 11:46:14 DEBUG Config - Config.get(instrumenting, swaps) =
2022-10-18 11:46:14 DEBUG Config - Config.get(instrumenting, compares)
2022-10-18 11:46:14 DEBUG Config - Config.get(instrumenting, inversion
2022-10-18 11:46:14 DEBUG Config - Config.get(instrumenting, fixes) =
2022-10-18 11:46:14 DEBUG Config - Config.get(instrumenting, hits) = t
2022-10-18 11:46:14 DEBUG Config - Config.get(helper, cutoff) =
Helper for InsertionSort with 4 elements
StatPack {hits: 9,880, normalized=21.454; copies: 0, normalized=0.000;
StatPack {hits: 19,800, normalized=42.995; copies: 0, normalized=0.000

Process finished with exit code 0
```

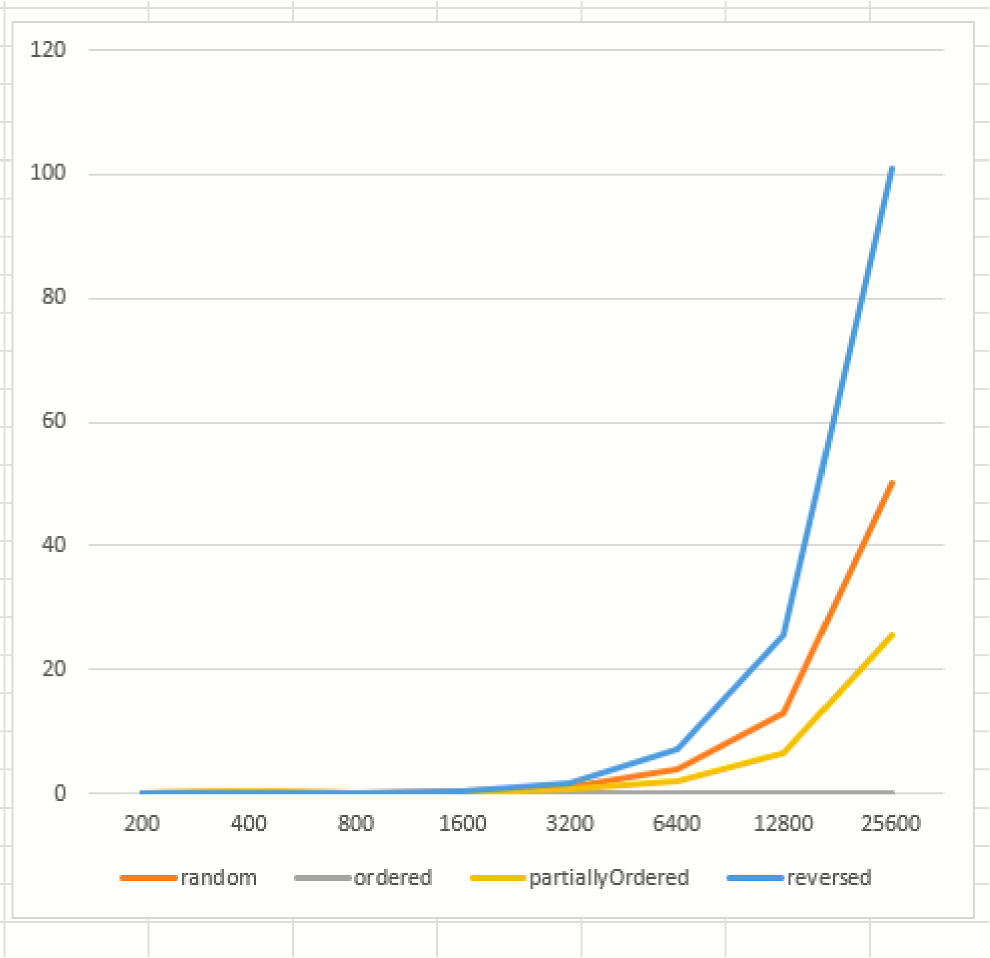
Part 3 Measure Running Times

Measured Data

Table:

n	random	ordered	partiallyOrdered	reversed
200	0.0374	0.00516	0.01614	0.04288
400	0.32443	0.00359	0.19526	0.05365
800	0.10849	0.0065	0.03351	0.11643
1600	0.25289	0.00811	0.12406	0.45562
3200	0.93144	0.01623	0.56324	1.79383
6400	3.84342	0.03391	1.83933	7.2329
12800	12.86309	0.05785	6.54417	25.50461
25600	50.28089	0.0905	25.56846	100.8463

Graph:



Obervation

I used doubling method to test the time usage of InsertionSort. For each different array length (n), I tried four array sorting situations (random, ordered, inverted, and partially ordered). Based on the output data and charts, we can briefly know that the speed of sorting ordered array is the fastest, sorting random and partially ordered arrray is slower, and sorting for reversed is the slowest.

For best situation (ordered), InsertionSort does not require a swap operation, it needs $(n-1)$ times comparison. For worest situation(reversed), InsertionSort needs $n*(n-1)/2$ times comparison, because when we insert nth element, we need to compare previous (n - 1) elements. InsertionSort's swap times is number of comparison operations minus $(n-1)$. On the average, the time complexity of the InsertionSort is $O(n^2)$.