Program Structures and Algorithms
Spring 2023(SEC 3)

NAME: Keerthana Satheesh
NUID: 002747795

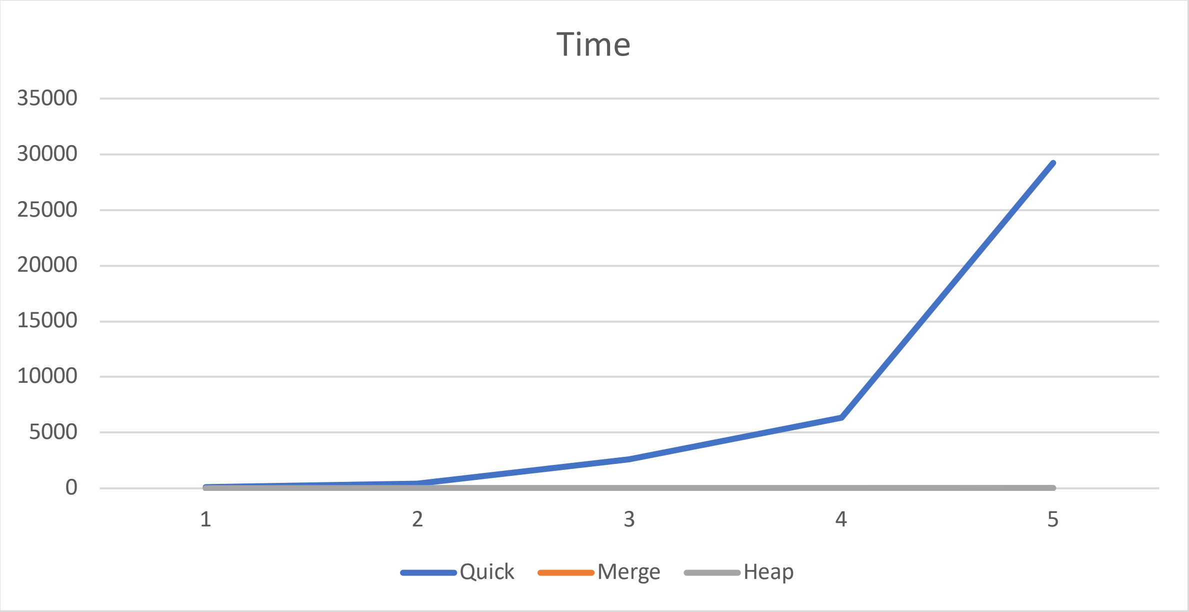
Task: determine--for sorting algorithms--what is the best predictor of total execution time: comparisons, swaps/copies, hits (array accesses)

Relationship Conclusion: From the graphs and values we can conclude that hits and compares are both important predictors of the total execution time for a sorting algorithm.

Evidence to support that conclusion:
Swaps are not good since merge sort always shows 0 for any number for array.
Hits and compare are good predictor as per analysis.

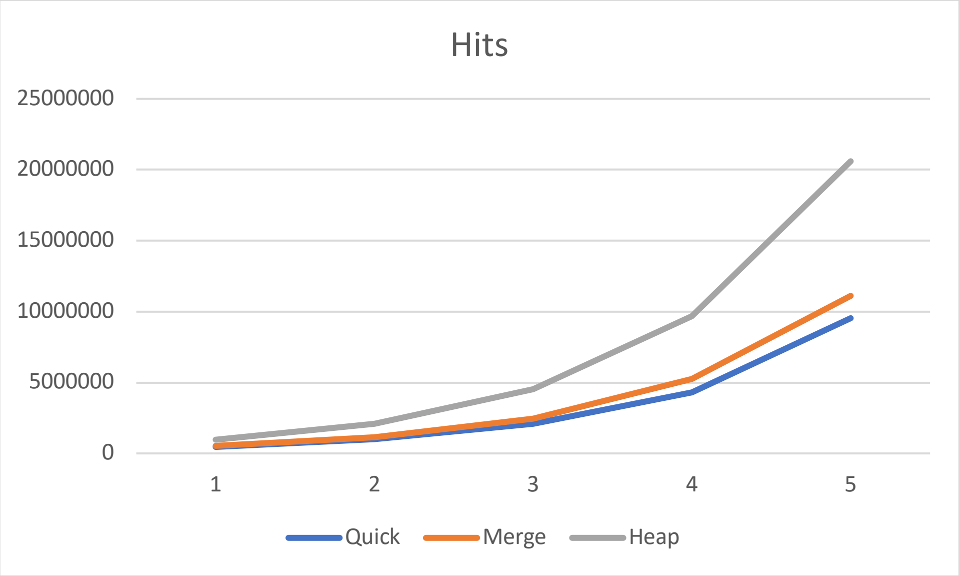
Time:-

Array Size	Quick	Merge	Heap
10000	109	2	1
20000	397	3	2
40000	2611	5	5
80000	6353	11	15
160000	29225	26	25



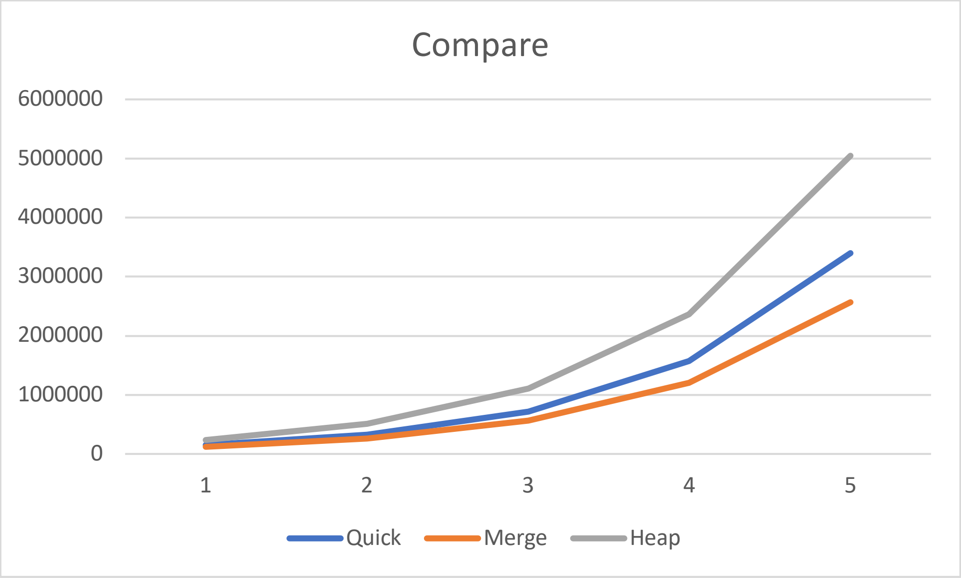
Hits:-

Array Size	Quick	Merge	Heap
10000	458902	534464	967602
20000	1003406	1148928	2095144
40000	2086888	2457856	4510436
80000	4320813	5235712	9661698
160000	9536891	11111424	20602072



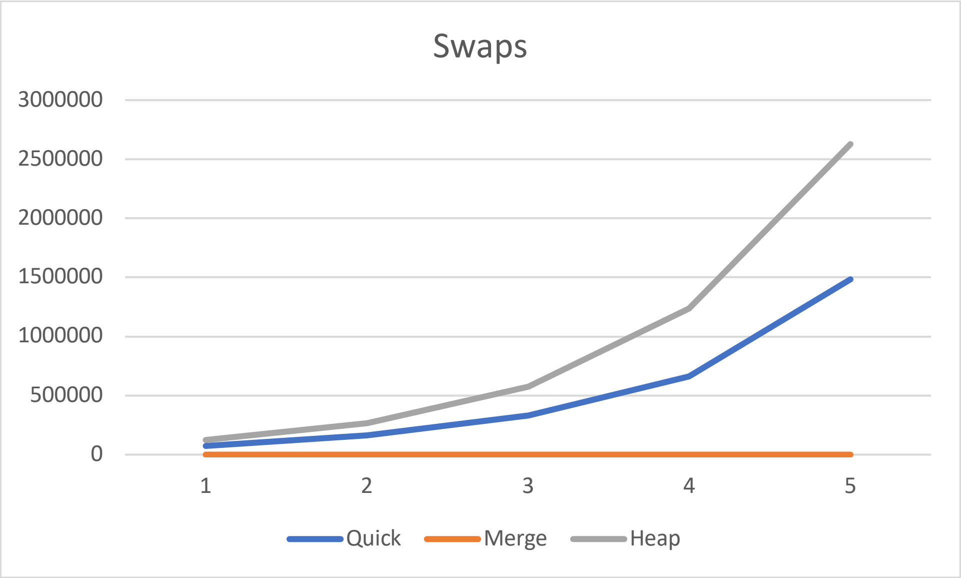
Compare:-

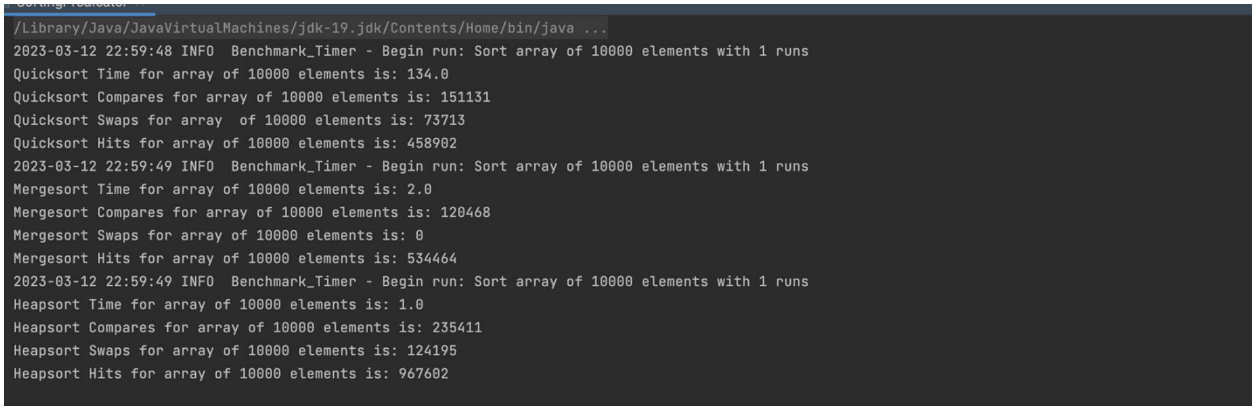
Array Size	Quick	Merge	Heap
10000	151131	120468	235411
20000	327462	260956	510710
40000	717101	561753	1101488
80000	1576435	1203457	2363059
160000	3398105	2567003	5046212



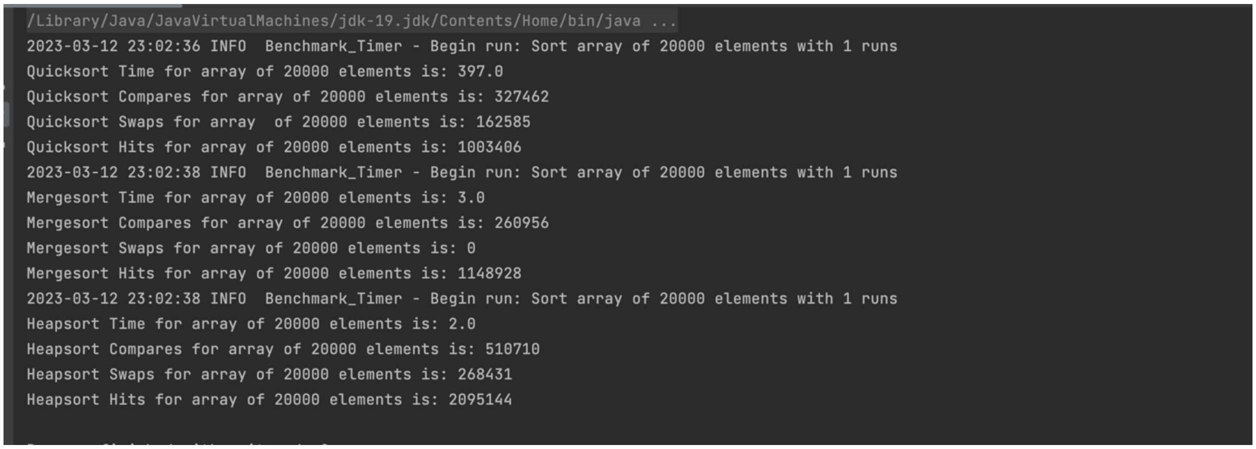
Swaps:-

Array Size	Quick	Merge	Heap
10000	73713	0	124195
20000	162585	0	268431
40000	329674	0	576865
80000	660473	0	1233895
160000	1483524	0	2627412

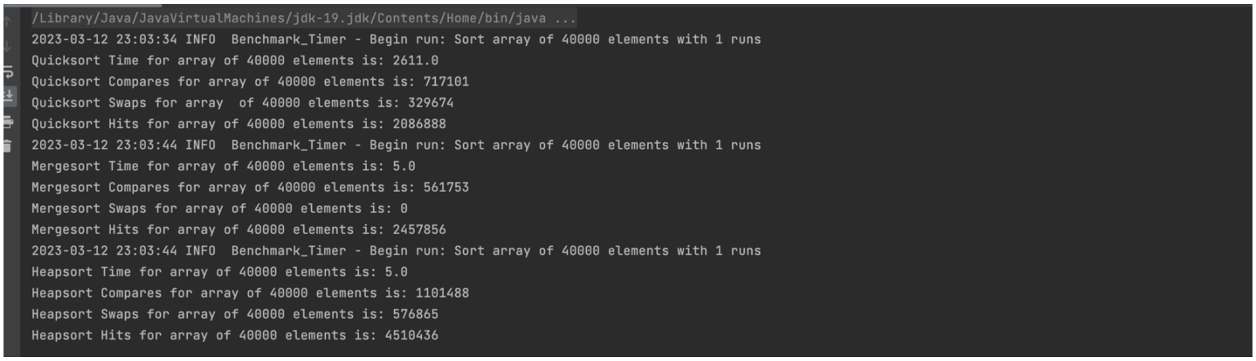




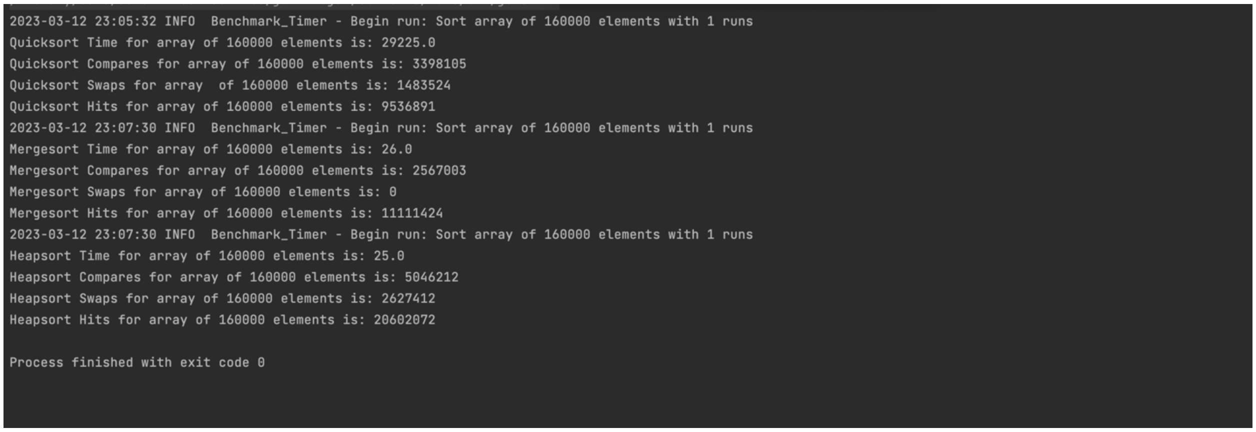
```
/Library/Java/JavaVirtualMachines/jdk-19.jdk/Contents/Home/bin/java ...
2023-03-12 22:59:48 INFO  Benchmark_Timer - Begin run: Sort array of 10000 elements with 1 runs
Quicksort Time for array of 10000 elements is: 134.0
Quicksort Compares for array of 10000 elements is: 151131
Quicksort Swaps for array  of 10000 elements is: 73713
Quicksort Hits for array of 10000 elements is: 458902
2023-03-12 22:59:49 INFO  Benchmark_Timer - Begin run: Sort array of 10000 elements with 1 runs
Mergesort Time for array of 10000 elements is: 2.0
Mergesort Compares for array of 10000 elements is: 120468
Mergesort Swaps for array of 10000 elements is: 0
Mergesort Hits for array of 10000 elements is: 534464
2023-03-12 22:59:49 INFO  Benchmark_Timer - Begin run: Sort array of 10000 elements with 1 runs
Heapsort Time for array of 10000 elements is: 1.0
Heapsort Compares for array of 10000 elements is: 235411
Heapsort Swaps for array of 10000 elements is: 124195
Heapsort Hits for array of 10000 elements is: 967602
```



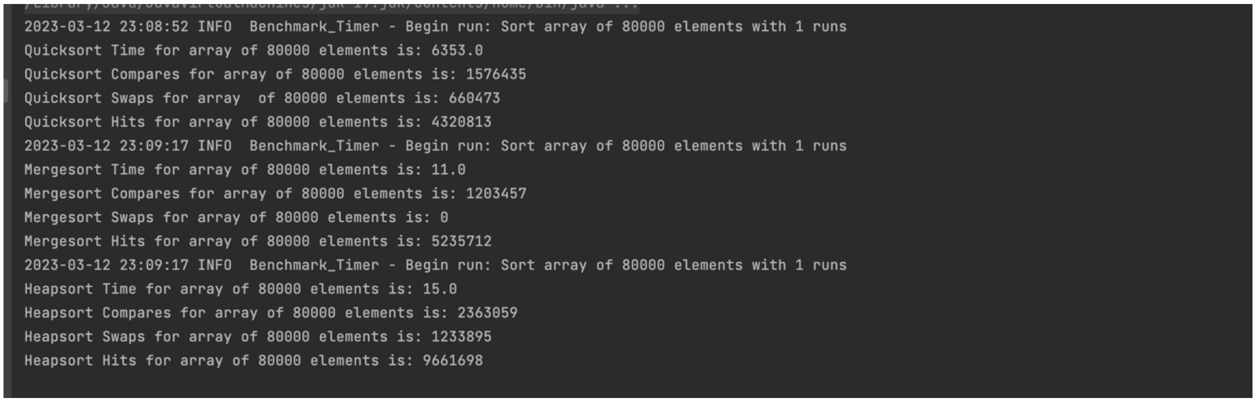
```
/Library/Java/JavaVirtualMachines/jdk-19.jdk/Contents/Home/bin/java ...
2023-03-12 23:02:36 INFO  Benchmark_Timer - Begin run: Sort array of 20000 elements with 1 runs
Quicksort Time for array of 20000 elements is: 397.0
Quicksort Compares for array of 20000 elements is: 327462
Quicksort Swaps for array  of 20000 elements is: 162585
Quicksort Hits for array of 20000 elements is: 1003406
2023-03-12 23:02:38 INFO  Benchmark_Timer - Begin run: Sort array of 20000 elements with 1 runs
Mergesort Time for array of 20000 elements is: 3.0
Mergesort Compares for array of 20000 elements is: 260956
Mergesort Swaps for array of 20000 elements is: 0
Mergesort Hits for array of 20000 elements is: 1148928
2023-03-12 23:02:38 INFO  Benchmark_Timer - Begin run: Sort array of 20000 elements with 1 runs
Heapsort Time for array of 20000 elements is: 2.0
Heapsort Compares for array of 20000 elements is: 510710
Heapsort Swaps for array of 20000 elements is: 268431
Heapsort Hits for array of 20000 elements is: 2095144
```



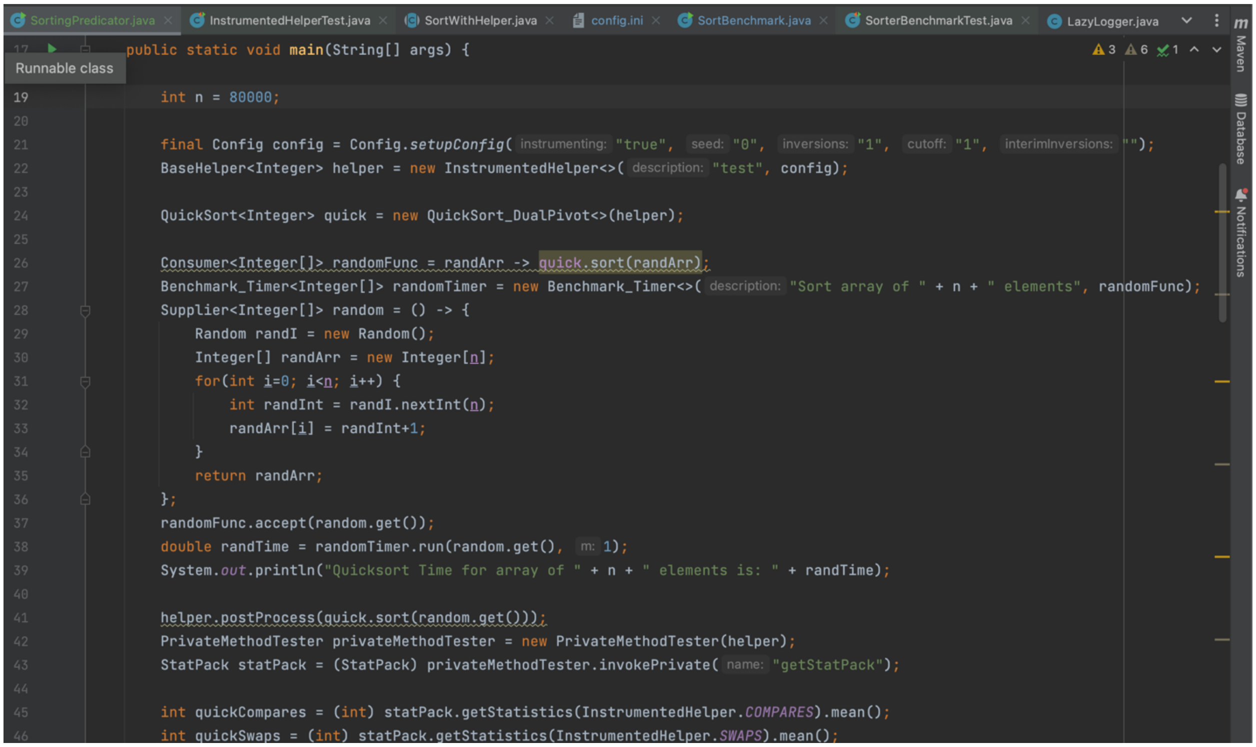
```
/Library/Java/JavaVirtualMachines/jdk-19.jdk/Contents/Home/bin/java ...
2023-03-12 23:03:34 INFO  Benchmark_Timer - Begin run: Sort array of 40000 elements with 1 runs
Quicksort Time for array of 40000 elements is: 2611.0
Quicksort Compares for array of 40000 elements is: 717101
Quicksort Swaps for array  of 40000 elements is: 329674
Quicksort Hits for array of 40000 elements is: 2086888
2023-03-12 23:03:44 INFO  Benchmark_Timer - Begin run: Sort array of 40000 elements with 1 runs
Mergesort Time for array of 40000 elements is: 5.0
Mergesort Compares for array of 40000 elements is: 561753
Mergesort Swaps for array of 40000 elements is: 0
Mergesort Hits for array of 40000 elements is: 2457856
2023-03-12 23:03:44 INFO  Benchmark_Timer - Begin run: Sort array of 40000 elements with 1 runs
Heapsort Time for array of 40000 elements is: 5.0
Heapsort Compares for array of 40000 elements is: 1101488
Heapsort Swaps for array of 40000 elements is: 576865
Heapsort Hits for array of 40000 elements is: 4510436
```



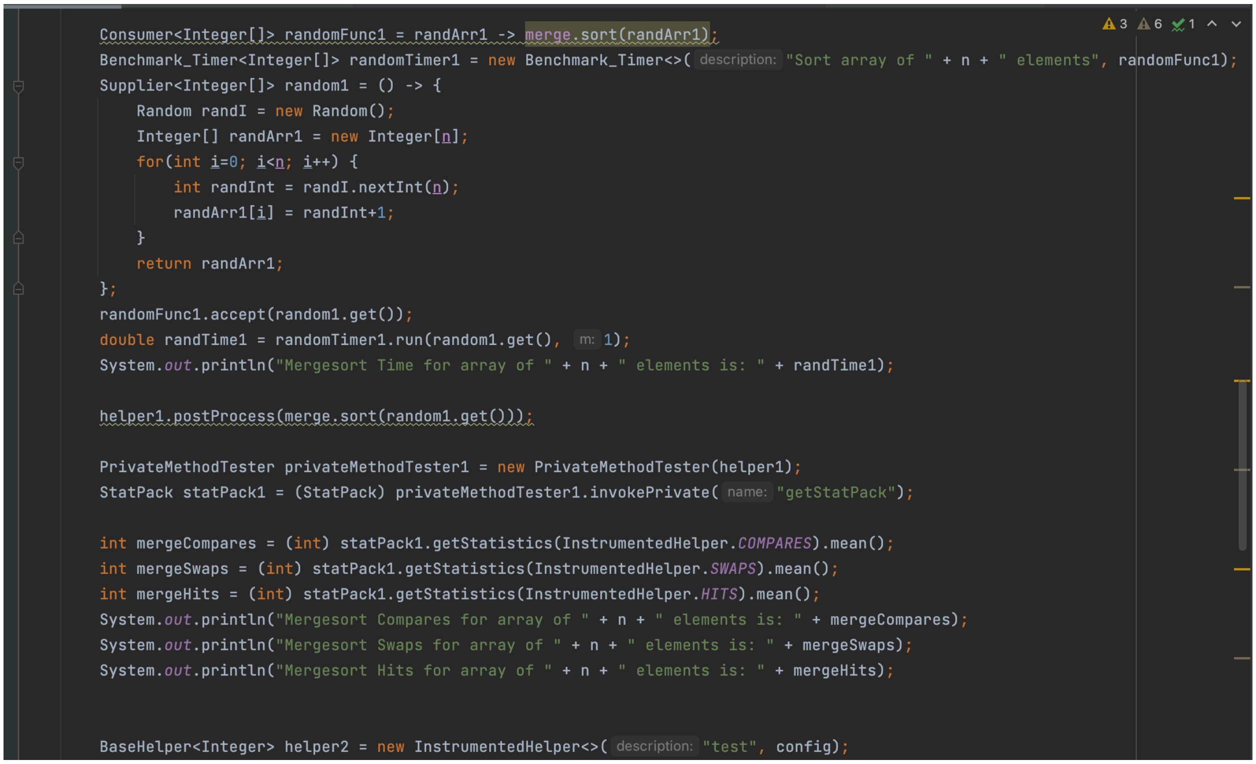
```
2023-03-12 23:05:32 INFO  Benchmark_Timer - Begin run: Sort array of 160000 elements with 1 runs
Quicksort Time for array of 160000 elements is: 29225.0
Quicksort Compares for array of 160000 elements is: 3398105
Quicksort Swaps for array  of 160000 elements is: 1483524
Quicksort Hits for array of 160000 elements is: 9536891
2023-03-12 23:07:30 INFO  Benchmark_Timer - Begin run: Sort array of 160000 elements with 1 runs
Mergesort Time for array of 160000 elements is: 26.0
Mergesort Compares for array of 160000 elements is: 2567003
Mergesort Swaps for array of 160000 elements is: 0
Mergesort Hits for array of 160000 elements is: 11111424
2023-03-12 23:07:30 INFO  Benchmark_Timer - Begin run: Sort array of 160000 elements with 1 runs
Heapsort Time for array of 160000 elements is: 25.0
Heapsort Compares for array of 160000 elements is: 5046212
Heapsort Swaps for array of 160000 elements is: 2627412
Heapsort Hits for array of 160000 elements is: 20602072

Process finished with exit code 0
```



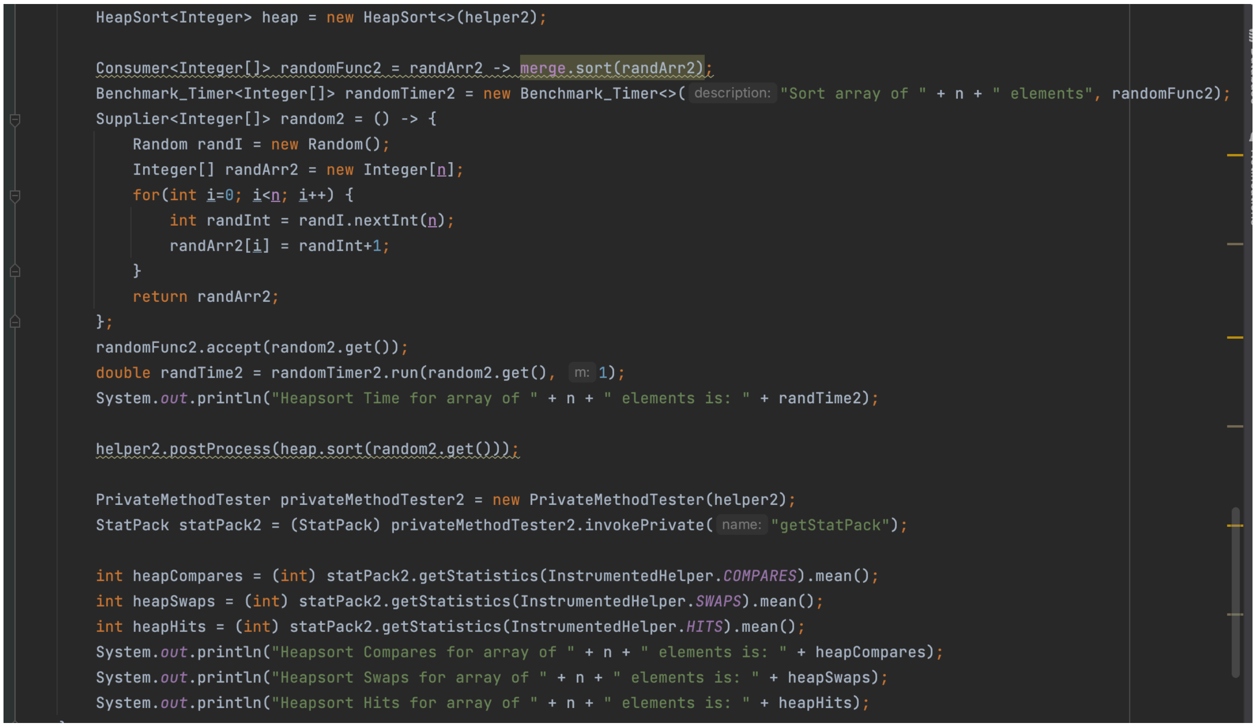
```
/Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/Home/bin/java ...
2023-03-12 23:08:52 INFO  Benchmark_Timer - Begin run: Sort array of 80000 elements with 1 runs
Quicksort Time for array of 80000 elements is: 6353.0
Quicksort Compares for array of 80000 elements is: 1576435
Quicksort Swaps for array  of 80000 elements is: 660473
Quicksort Hits for array of 80000 elements is: 4320813
2023-03-12 23:09:17 INFO  Benchmark_Timer - Begin run: Sort array of 80000 elements with 1 runs
Mergesort Time for array of 80000 elements is: 11.0
Mergesort Compares for array of 80000 elements is: 1203457
Mergesort Swaps for array of 80000 elements is: 0
Mergesort Hits for array of 80000 elements is: 5235712
2023-03-12 23:09:17 INFO  Benchmark_Timer - Begin run: Sort array of 80000 elements with 1 runs
Heapsort Time for array of 80000 elements is: 15.0
Heapsort Compares for array of 80000 elements is: 2363059
Heapsort Swaps for array of 80000 elements is: 1233895
Heapsort Hits for array of 80000 elements is: 9661698
```



```java
public static void main(String[] args) {

    int n = 80000;

    final Config config = Config.setupConfig( instrumenting: "true",  seed: "0",  inversions: "1",  cutoff: "1",  interimInversions: "");
    BaseHelper<Integer> helper = new InstrumentedHelper<>( description: "test", config);

    QuickSort<Integer> quick = new QuickSort_DualPivot<>(helper);

    Consumer<Integer[]> randomFunc = randArr -> quick.sort(randArr);
    Benchmark_Timer<Integer[]> randomTimer = new Benchmark_Timer<>( description: "Sort array of " + n + " elements", randomFunc);
    Supplier<Integer[]> random = () -> {
        Random randI = new Random();
        Integer[] randArr = new Integer[n];
        for(int i=0; i<n; i++) {
            int randInt = randI.nextInt(n);
            randArr[i] = randInt+1;
        }
        return randArr;
    };
    randomFunc.accept(random.get());
    double randTime = randomTimer.run(random.get(),  m: 1);
    System.out.println("Quicksort Time for array of " + n + " elements is: " + randTime);

    helper.postProcess(quick.sort(random.get()));
    PrivateMethodTester privateMethodTester = new PrivateMethodTester(helper);
    StatPack statPack = (StatPack) privateMethodTester.invokePrivate( name: "getStatPack");

    int quickCompares = (int) statPack.getStatistics(InstrumentedHelper.COMPARES).mean();
    int quickSwaps = (int) statPack.getStatistics(InstrumentedHelper.SWAPS).mean();
```



```java
Consumer<Integer[]> randomFunc1 = randArr1 -> merge.sort(randArr1);
Benchmark_Timer<Integer[]> randomTimer1 = new Benchmark_Timer<>( description: "Sort array of " + n + " elements", randomFunc1);
Supplier<Integer[]> random1 = () -> {
    Random randI = new Random();
    Integer[] randArr1 = new Integer[n];
    for(int i=0; i<n; i++) {
        int randInt = randI.nextInt(n);
        randArr1[i] = randInt+1;
    }
    return randArr1;
};
randomFunc1.accept(random1.get());
double randTime1 = randomTimer1.run(random1.get(),  m: 1);
System.out.println("Mergesort Time for array of " + n + " elements is: " + randTime1);

helper1.postProcess(merge.sort(random1.get()));

PrivateMethodTester privateMethodTester1 = new PrivateMethodTester(helper1);
StatPack statPack1 = (StatPack) privateMethodTester1.invokePrivate( name: "getStatPack");

int mergeCompares = (int) statPack1.getStatistics(InstrumentedHelper.COMPARES).mean();
int mergeSwaps = (int) statPack1.getStatistics(InstrumentedHelper.SWAPS).mean();
int mergeHits = (int) statPack1.getStatistics(InstrumentedHelper.HITS).mean();
System.out.println("Mergesort Compares for array of " + n + " elements is: " + mergeCompares);
System.out.println("Mergesort Swaps for array of " + n + " elements is: " + mergeSwaps);
System.out.println("Mergesort Hits for array of " + n + " elements is: " + mergeHits);


BaseHelper<Integer> helper2 = new InstrumentedHelper<>( description: "test", config);
```



```
        HeapSort<Integer> heap = new HeapSort<>(helper2);

        Consumer<Integer[]> randomFunc2 = randArr2 -> merge.sort(randArr2);
        Benchmark_Timer<Integer[]> randomTimer2 = new Benchmark_Timer<>( description: "Sort array of " + n + " elements", randomFunc2);
        Supplier<Integer[]> random2 = () -> {
            Random randI = new Random();
            Integer[] randArr2 = new Integer[n];
            for(int i=0; i<n; i++) {
                int randInt = randI.nextInt(n);
                randArr2[i] = randInt+1;
            }
            return randArr2;
        };
        randomFunc2.accept(random2.get());
        double randTime2 = randomTimer2.run(random2.get(),  m: 1);
        System.out.println("Heapsort Time for array of " + n + " elements is: " + randTime2);

        helper2.postProcess(heap.sort(random2.get()));

        PrivateMethodTester privateMethodTester2 = new PrivateMethodTester(helper2);
        StatPack statPack2 = (StatPack) privateMethodTester2.invokePrivate( name: "getStatPack");

        int heapCompares = (int) statPack2.getStatistics(InstrumentedHelper.COMPARES).mean();
        int heapSwaps = (int) statPack2.getStatistics(InstrumentedHelper.SWAPS).mean();
        int heapHits = (int) statPack2.getStatistics(InstrumentedHelper.HITS).mean();
        System.out.println("Heapsort Compares for array of " + n + " elements is: " + heapCompares);
        System.out.println("Heapsort Swaps for array of " + n + " elements is: " + heapSwaps);
        System.out.println("Heapsort Hits for array of " + n + " elements is: " + heapHits);
```