```java
package edu.neu.coe.info6205.sort.elementary;

import edu.neu.coe.info6205.sort.Helper;
import edu.neu.coe.info6205.sort.InstrumentedHelper;
import edu.neu.coe.info6205.sort.SortWithHelper;
import edu.neu.coe.info6205.util.Benchmark;
import edu.neu.coe.info6205.util.Benchmark_Timer;
import edu.neu.coe.info6205.util.Config;

public class HeapSort<X extends Comparable<X>> extends SortWithHelper<X> {

    public HeapSort(Helper<X> helper) {
        super(helper);
    }

    @Override
    public void sort(X[] array, int from, int to) {
        if (array == null || array.length <= 1) return;

        // XXX construction phase
        buildMaxHeap(array);

        // XXX sort-down phase
        Helper<X> helper = getHelper();
        for (int i = array.length - 1; i >= 1; i--) {
            helper.swap(array, 0, i);
            maxHeap(array, i, 0);
        }
    }

    private void buildMaxHeap(X[] array) {
        int half = array.length / 2;
        for (int i = half; i >= 0; i--) maxHeap(array, array.length, i);
    }

    private void maxHeap(X[] array, int heapSize,
```

```
36 int index) {
37         Helper<X> helper = getHelper();
38         final int left = index * 2 + 1;
39         final int right = index * 2 + 2;
40         int largest = index;
41         if (left < heapSize && helper.compare(array
   , largest, left) < 0) largest = left;
42         if (right < heapSize && helper.compare(
   array, largest, right) < 0) largest = right;
43         if (index != largest) {
44             helper.swap(array, index, largest);
45             maxHeap(array, heapSize, largest);
46         }
47     }
48
49     public static void main(String[] args) {
50         int N = 10000;
51
52         for(int i=2;i<12;i++) {
53             InstrumentedHelper<Integer>
   instrumentedHelper = new InstrumentedHelper<>("
   HeapSort", Config.setupConfig("true", "0", "1", ""
   , ""));
54             HeapSort<Integer> s = new HeapSort<>(
   instrumentedHelper);
55             int j = N * i / 2;
56             s.init(j);
57             Integer[] temp = instrumentedHelper.
   random(Integer.class, r -> r.nextInt(j));
58             Benchmark<Boolean> benchmark = new
   Benchmark_Timer<>("Sorting with", b -> s.sort(temp
   , 0, j));
59             double x = benchmark.run(true, 20);
60             s.sort(temp, 0, j);
61
62             long nCompares = instrumentedHelper.
   getCompares();
63             int nSwaps = instrumentedHelper.
   getSwaps();
64             int nHits = instrumentedHelper.getHits
   ();
```

```
65
66              System.out.println("When array size is
    : " + j);
67              System.out.println("Compares: " +
    nCompares);
68              System.out.println("Swaps: " + nSwaps
     );
69              System.out.println("hits: " + nHits);
70              System.out.println("Time: " + x);
71
72              System.out.println("\n\n");
73          }
74      }
75  }
```