```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,
int index) {
        Helper<X> helper = getHelper();
        final int left = index * 2 + 1;
        final int right = index * 2 + 2;
        int largest = index;
        if (left < heapSize && helper.compare(array, largest, left) < 0) largest = left;
        if (right < heapSize && helper.compare(array, largest, right) < 0) largest = right;
        if (index != largest) {
            helper.swap(array, index, largest);
            maxHeap(array, heapSize, largest);
        }
    }

    public static void main(String[] args) {
        int N = 1000;

        while(N<=64000) {
            InstrumentedHelper<Integer> instrumentedHelper = new InstrumentedHelper<>("HeapSort", Config.setupConfig("true", "0", "1", "", ""));
            HeapSort<Integer> s = new HeapSort<>(instrumentedHelper);
            int j = N;
            s.init(j);
            Integer[] temp = instrumentedHelper.random(Integer.class, r -> r.nextInt(j));
            Benchmark<Boolean> benchmark = new Benchmark_Timer<>("Sorting", b -> s.sort(temp, 0, j));
            double nTime = benchmark.run(true, 20);
            s.sort(temp, 0, j);

            long nCompares = instrumentedHelper.getCompares();
            int nSwaps = instrumentedHelper.getSwaps();
            int nHits = instrumentedHelper.getHits();

            System.out.println("When array size is : " + j);
            System.out.println("Compares: " + nCompares);
            System.out.println("Swaps: " + nSwaps);
            System.out.println("Hits: " + nHits);
            System.out.println("Time: " + nTime);

            System.out.println("\nFor referencs:\t" + j + "\t" + nCompares + "\t" + nSwaps + "\t" + nHits + "\t" + nTime + "\n");

            N = N*2;
        }
    }
}
```