```java
/**
 * COMP215-Programming Project 2: Multiple Sort Analysis.
 * MAINAPP holds the main() method, which runs algorithm analyses.
 * The main() method increments a dataset size and runs sorts on each dataset.
 * @author Andrew Parsons
 * @version 06 March 2017
 */
public class MainApp {

    static boolean debug = false;
    static MultiFileWriter multiFileWriter = new MultiFileWriter();

    public static void main(String args[]) {

        /** --- SET VALUES HERE ------------------------------------------------------------- */
        int maxValue = 1000;    // this sets the largest element value in an array
        int repetitions = 5;   // this sets the number of repetitions for the mean calculation
        int startArraySize = 0;    // this sets the starting array's size
        int endArraySize = 100000;   // this sets the ending array's size
        int increment = 1000;   // this sets the increment size
        /** --------------------------------------------------------------------------------- */

        // instantiate the necessary objects: a data generator and the four sorters
        DataGenerator dataGenerator = new DataGenerator();
        InsertionSort insertionSort = new InsertionSort();
        MergeSort mergeSort = new MergeSort();
        HeapSort heapSort = new HeapSort();
        QuickSort quickSort = new QuickSort();

        // run the sorting algorithm on datasets of increasing size
        for (int dataSize = startArraySize; dataSize < (endArraySize + increment); dataSize =
        dataSize + increment) {

            Comparable[] dataset = dataGenerator.createDataSet(dataSize, maxValue);

            for (int algToTest = 0; algToTest < 4; algToTest++) {
                switch (algToTest) {
                    case 0:
                        // insertion sort testing
                        AlgorithmTester algorithmTester = new AlgorithmTester(insertionSort);
                        algorithmTester.testAlgorithm(dataset, repetitions);
                        System.out.println();
                        break;
                    case 1:
                        // merge sort testing
                        algorithmTester = new AlgorithmTester(mergeSort);
                        algorithmTester.testAlgorithm(dataset, repetitions);
                        System.out.println();
                        break;
                    case 2:
                        // heap sort testing
                        algorithmTester = new AlgorithmTester(heapSort);
                        algorithmTester.testAlgorithm(dataset, repetitions);
                        System.out.println();
                        break;
                    case 3:
                        // quick sort testing
                        algorithmTester = new AlgorithmTester(quickSort);
                        algorithmTester.testAlgorithm(dataset, repetitions);
                        System.out.println();
                        break;
                }
            }
        }
```

```
    }
}
```