```
import java.io.FileWriter;
import java.io.IOException;
import java.util.ArrayList;

/**
 * COMP215-Programming Project 2: Multiple Sort Analysis.
 * MULTIFILEWRITER is a wrapper for 12 other FileWriters.
 * TestResults are parsed and directed to their proper destination.
 * @author Andrew Parsons
 * @version 05 March 2017
 *
 */
class MultiFileWriter {

    /* --- INSTANCE VARIABLES --- */

    private FileWriter insertion_random;
    private FileWriter insertion_ascending;
    private FileWriter insertion_descending;

    private FileWriter merge_random;
    private FileWriter merge_ascending;
    private FileWriter merge_descending;

    private FileWriter heap_random;
    private FileWriter heap_ascending;
    private FileWriter heap_descending;

    private FileWriter quick_random;
    private FileWriter quick_ascending;
    private FileWriter quick_descending;

    /* --- METHODS --- */

    /**
     * (package-private) MULTIFILEWRITER is the constructor for this object.
     * The constructor instantiates an ArrayList of FileWriters, sets their output directories,
     * creates each file, and writes the CSV header row.
     */
    MultiFileWriter() {

        ArrayList<FileWriter> fileWriters = new ArrayList<>();

        String insertionDIR = "output/insertion/";
        String mergeDIR = "output/merge/";
        String heapDIR = "output/heap/";
        String quickDIR = "output/quick/";
        String random = "random.csv";
        String ascending = "ascending.csv";
        String descending = "descending.csv";

        try {
            insertion_random = new FileWriter(insertionDIR + random);
            insertion_ascending = new FileWriter(insertionDIR + ascending);
            insertion_descending = new FileWriter(insertionDIR + descending);

            merge_random = new FileWriter(mergeDIR + random);
            merge_ascending = new FileWriter(mergeDIR + ascending);
            merge_descending = new FileWriter(mergeDIR + descending);

            heap_random = new FileWriter(heapDIR + random);
            heap_ascending = new FileWriter(heapDIR + ascending);
            heap_descending = new FileWriter(heapDIR + descending);
```

```
            quick_random = new FileWriter(quickDIR + random);
            quick_ascending = new FileWriter(quickDIR + ascending);
            quick_descending = new FileWriter(quickDIR + descending);

            fileWriters.add(insertion_random);
            fileWriters.add(insertion_ascending);
            fileWriters.add(insertion_descending);

            fileWriters.add(merge_random);
            fileWriters.add(merge_ascending);
            fileWriters.add(merge_descending);

            fileWriters.add(heap_random);
            fileWriters.add(heap_ascending);
            fileWriters.add(heap_descending);

            fileWriters.add(quick_random);
            fileWriters.add(quick_ascending);
            fileWriters.add(quick_descending);
        } catch (IOException e) {
            e.printStackTrace();
        }

        for (FileWriter fileWriter : fileWriters) {
            try {
                fileWriter.append("n");
                fileWriter.append(',');
                fileWriter.append("T(n)");
                fileWriter.append('\n');
                fileWriter.flush();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }

    /**
     * (package-private) PROCESSTESTRESULT directs an incoming TestResult to the correct
     FileWriter, where it is parsed
     * and written to the CSV file.
     * @param testResult the incoming TestResult from AlgorithmTester
     * @param sorter the sorter tested in AlgorithmTester
     * @param sortType the type of dataset (random, ascendingly sorted, descendingly sorted)
     * @throws Exception
     */
    void processTestResult(TestResult testResult, Sorter sorter, String sortType) throws
    Exception {

       if (sorter instanceof InsertionSort) {
            if (sortType.equals("random")) {
                insertion_random.append(testResult.returnSizeOfArrayAsString());
                insertion_random.append(',');
                insertion_random.append(testResult.returnMeanTimeAsDoubleString());
                insertion_random.append('\n');
                insertion_random.flush();
            }

            else if (sortType.equals("ascending")) {
                insertion_ascending.append(testResult.returnSizeOfArrayAsString());
                insertion_ascending.append(',');
                insertion_ascending.append(testResult.returnMeanTimeAsDoubleString());
                insertion_ascending.append('\n');
                insertion_ascending.flush();
            }
```

```
        else if (sortType.equals("descending")) {
            insertion_descending.append(testResult.returnSizeOfArrayAsString());
            insertion_descending.append(',');
            insertion_descending.append(testResult.returnMeanTimeAsDoubleString());
            insertion_descending.append('\n');
            insertion_descending.flush();
        }

        else {
            System.out.println("unidentifiable sortType in
            MultiFileWriter.processTestResult()!");
            throw new Exception();
        }
    }

     else if (sorter instanceof MergeSort) {
        if (sortType.equals("random")) {
            merge_random.append(testResult.returnSizeOfArrayAsString());
            merge_random.append(',');
            merge_random.append(testResult.returnMeanTimeAsDoubleString());
            merge_random.append('\n');
            merge_random.flush();
        }

        else if (sortType.equals("ascending")) {
            merge_ascending.append(testResult.returnSizeOfArrayAsString());
            merge_ascending.append(',');
            merge_ascending.append(testResult.returnMeanTimeAsDoubleString());
            merge_ascending.append('\n');
            merge_ascending.flush();
        }

        else if (sortType.equals("descending")) {
            merge_descending.append(testResult.returnSizeOfArrayAsString());
            merge_descending.append(',');
            merge_descending.append(testResult.returnMeanTimeAsDoubleString());
            merge_descending.append('\n');
            merge_descending.flush();
        }

        else {
            System.out.println("unidentifiable sortType in
            MultiFileWriter.processTestResult()!");
            throw new Exception();
        }
     }

    else if (sorter instanceof HeapSort) {
        if (sortType.equals("random")) {
            heap_random.append(testResult.returnSizeOfArrayAsString());
            heap_random.append(',');
            heap_random.append(testResult.returnMeanTimeAsDoubleString());
            heap_random.append('\n');
            heap_random.flush();
        }

        else if (sortType.equals("ascending")) {
            heap_ascending.append(testResult.returnSizeOfArrayAsString());
            heap_ascending.append(',');
            heap_ascending.append(testResult.returnMeanTimeAsDoubleString());
            heap_ascending.append('\n');
            heap_ascending.flush();
        }
```

```
            else if (sortType.equals("descending")) {
                heap_descending.append(testResult.returnSizeOfArrayAsString());
                heap_descending.append(',');
                heap_descending.append(testResult.returnMeanTimeAsDoubleString());
                heap_descending.append('\n');
                heap_descending.flush();
            }

            else {
                System.out.println("Unidentifiable sortType in
                MultiFileWriter.processTestResult()!");
                throw new Exception();
            }
        }

        else if (sorter instanceof QuickSort) {
            if (sortType.equals("random")) {
                quick_random.append(testResult.returnSizeOfArrayAsString());
                quick_random.append(',');
                quick_random.append(testResult.returnMeanTimeAsDoubleString());
                quick_random.append('\n');
                quick_random.flush();
            }

            else if (sortType.equals("ascending")) {
                quick_ascending.append(testResult.returnSizeOfArrayAsString());
                quick_ascending.append(',');
                quick_ascending.append(testResult.returnMeanTimeAsDoubleString());
                quick_ascending.append('\n');
                quick_ascending.flush();
            }

            else if (sortType.equals("descending")) {
                quick_descending.append(testResult.returnSizeOfArrayAsString());
                quick_descending.append(',');
                quick_descending.append(testResult.returnMeanTimeAsDoubleString());
                quick_descending.append('\n');
                quick_descending.flush();
            }

            else {
                System.out.println("unidentifiable sortType in
                MultiFileWriter.processTestResult()!");
                throw new Exception();
            }
        }

        else {
            System.out.println("Hmmm...unknown sorting algorithm! Unsure of where to direct
            output!");
            throw new Exception();
        }
    }
}
```