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

/**
 * COMP215-Programming Project 3: 01 Knapsack Problem.
 * MULTIFILEWRITER is a wrapper for 3 other FileWriters.
 * TestResults are parsed and directed to their proper destination.
 * @author Andrew Parsons
 * @version 08 March 2017
 */
class MultiFileWriter {

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

    private FileWriter greedy;
    private FileWriter dynamic;
    private FileWriter bruteforce;

    /* --- 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 greedyDIR = "output/greedy.csv";
        String dynamicDIR = "output/dynamic.csv";
        String bruteforceDIR = "output/bruteforce.csv";

        try {
            greedy = new FileWriter(greedyDIR);
            dynamic = new FileWriter(dynamicDIR);
            bruteforce = new FileWriter(bruteforceDIR);
            fileWriters.add(greedy);
            fileWriters.add(dynamic);
            fileWriters.add(bruteforce);

        } 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 solution the solution tested in AlgorithmTester
     * @throws Exception
     */
    void processTestResult(TestResult testResult, Solution solution) throws Exception {

       if (solution instanceof GreedySolution) {
           greedy.append(testResult.returnSizeOfArrayAsString());
           greedy.append(',');
           greedy.append(testResult.returnMeanTimeAsDoubleString());
           greedy.append('\n');
           greedy.flush();
       }

       else if (solution instanceof DynamicSolution) {
           dynamic.append(testResult.returnSizeOfArrayAsString());
           dynamic.append(',');
           dynamic.append(testResult.returnMeanTimeAsDoubleString());
           dynamic.append('\n');
           dynamic.flush();
       }

       else if (solution instanceof BruteforceSolution) {
           bruteforce.append(testResult.returnSizeOfArrayAsString());
           bruteforce.append(',');
           bruteforce.append(testResult.returnMeanTimeAsDoubleString());
           bruteforce.append('\n');
           bruteforce.flush();
       }
    }
}
```