**Considerations and assumptions**

* It was not clear to me what to do with the IPs in the addresses file. I understood that it may have something to do with ensuring that proxies invoke the third party using a different IP to increase throughput, but I did not see how to make the proxy do that. So I had a single proxy running on port 8080 and I invoked it using http://localhost:8080/api/data?input=abc
* The only thing that I could make concurrent is the streaming of the input file and invoking the proxy. The writing of the output is using BufferedWriter which is synchronized. The object’s state is the:
  + The input and address file contents: These are populated before the concurrent operation and is not modified, so a HashMap is used.
  + A set of items that failed with 503. This is written in a concurrent context and a thread-safe data structure is used.
* The JUnit test is not a “unit test”. It requires the proxy to be running and it runs the main method.
* Before running, please ensure that no file exists for the output file. The program appends to the file if it exists.

**How to test:**

* For convenience, the input and address files are in src\main\resources. The JUnit test read these files; however, they can be supplied as runtime arguments to a different file if running manually.
* The test file is in src\test\java\TestMainClass.java. If using IntelliJ, right click and run this will execute the test.
* The test assumes that for a given input, each invocation of the proxy will return the same result. That has been the case for all the tests that I did so for assertion purposes, I have an expected output file saved in src\test\resources\expectedOutputfile.txt and assert that is equal to the results returned by the test.
* To test manually from an IDE without executing the JUnit test, ensure that the arguments are supplied e.g. -inputFile=src/main/resources/inputfile.txt -addressFile=src/main/resources/addressfile.txt -outputfile=src/main/resources/outputfile.txt