# Advantest Assignment

# Document Information

|  |  |
| --- | --- |
| Author(S) | Shashwat Anand |
| Version | 1.0 |
| Date | 10th March 2019 |

# 1. Business Summary

## 

## 1.1 Objective

Design a tester tool which can read test suites and execute them on test system depending on required OS, devices and report the status of execution.

## 1.2 Design Abstract

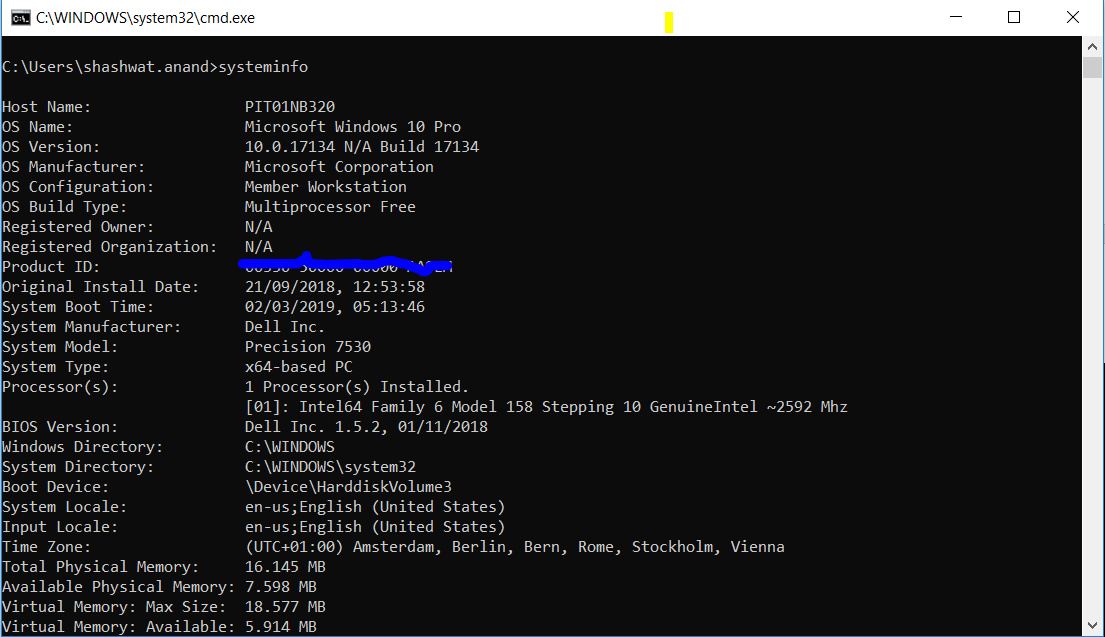
As part of this requirement, following things has to be implemented: -

1. Retrieve the stored test suite files from shared drive (for assignment I am reading from folder)
2. Ideally tester tool which read and figure out the required OS and devices need for execution of the test suite (for POC I kept the name of files as TSuite\_mac\_D1\_D2\_19, where test suite is “TSuite19”, required OS is “mac”, D1 and D2 are devices which is needed to execute the test suite and 19 is the execution time of it.
3. When user passes the test suite file name as argument then tester tool finds the its required OS, needed devices.
4. Then tester tool finds the test system which is having same OS and devices and currently not in use. Ideally this information can be stored in some data base and retrieved (for POC I didn’t created data base schema or entities due to time constraints, I just tried to model the test system data with POJO and keep it in in-memory data structure.
5. Once tester tool finds the test system the it should start the execution of test suite and return to process other test suite which is passed as program arguments to it.
6. Each test suite execution should report its execution status on completion to the user who has started it. To retrieve email is ideally we should read the user name of from system variables and query LDAP (active directory) or company database to find user email and send the report to him/her (here in POC is retrieved the user name used hard code email id to send the report)

# 2. System

# 2.1 System Configuration

Below is my system configuration.



# 2.2 Design

Input Console or File

(Java NIO for File)

(UI Eclipse Plugin)

Controller/Search Module

Output in Console/File

Testing

(Junit)

UI testing

(Manually)

# 3. Benefits

1. UI application
2. Colour Coding used for showing error or success in search results
3. UFX

# 4. Trade Off/Assumption

Assumption

1. Application will be tested in windows 64 bit machine
2. JDK 8 is used for developing the application so application should only tested on machines where JRE 8 is installed.
   1. Will also provide bundled jre in executables (but can’t test this fully as lack of environments)
3. Eclipse Neon is used as runtime for the application, which can be changed with simple configuration in POM file.
4. Tuples format is - ID, “Name”, Pattern, flag
   1. 1, “MyPattern”, src/patterns/Functional1.pat, false
   2. 2, “myPattern”, src/patterns/Functional2.pat, true
5. All search queries are case-sensitive.
6. In dumping queries into file functionality, I am writing the whole file (not appending it). Also I added \r\n for line break, so user of the system will have proper output of file in windows notepad.

Trade Off

As requirement document say tuples need to be in memory. I consider my development machine as the minimum-maximum sustainable system.

# 5. Improvement

1. Giving support to search using wild cards.
2. UI testing can be automated.
3. Adding support and testing for different platforms

# 6. Time Complexity

As far is time complexity is concerned for queries is Big O (n)

As you can see in class SearchResultProvider, I used stream and parallel stream. Stream filter always traverses the complete and applies the filter criteria to each element.

I used parallel stream only when if number of available processor is more than 1 and number of tuples in memory as more than 1000. Parallel stream give better performance if it has to act on more data to process.

# 7. Remark

Design took maximum time, had to take lot decision

1. UI or Console Application
2. Should I use functional paradigm of Java

8. Load Analysis

With all my day today applications, I found the following results. When I loaded the application with 300 MB text file with 129 processes running on system (Fig 8.1). Following are the observations.

* System was using 5.75 GB of RAM (Fig 8.2)
* After loading the 300 MB file RAM utilization went up by 700 MB (approx.) (Fig 8.3).
* It took 5 seconds to load the file “Time take to load file : 4820” ms

To summarize if we want to load 1 GB file then we may require 2.5 GB RAM for loading the file with you day to day application.

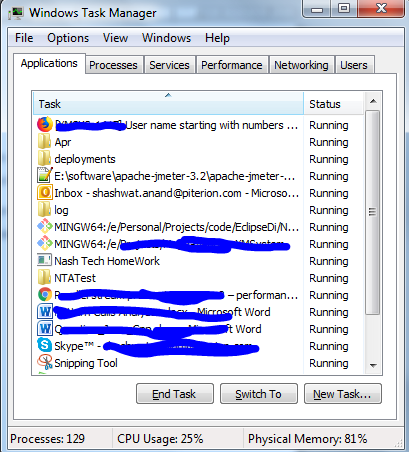


Fig 8. 1

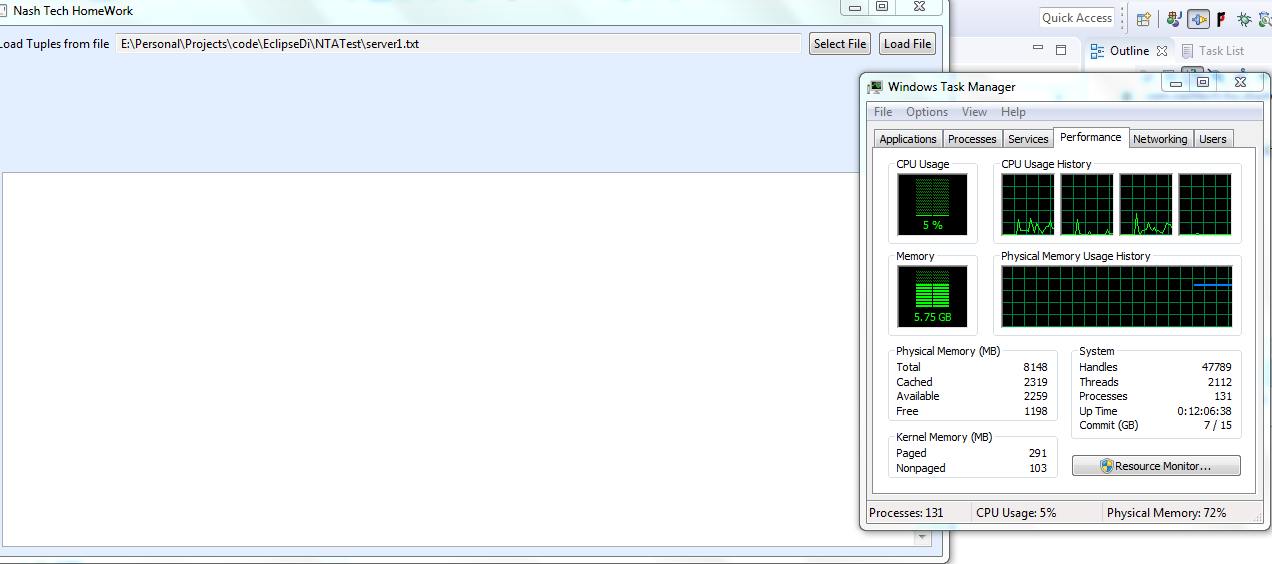


Fig 8. 2

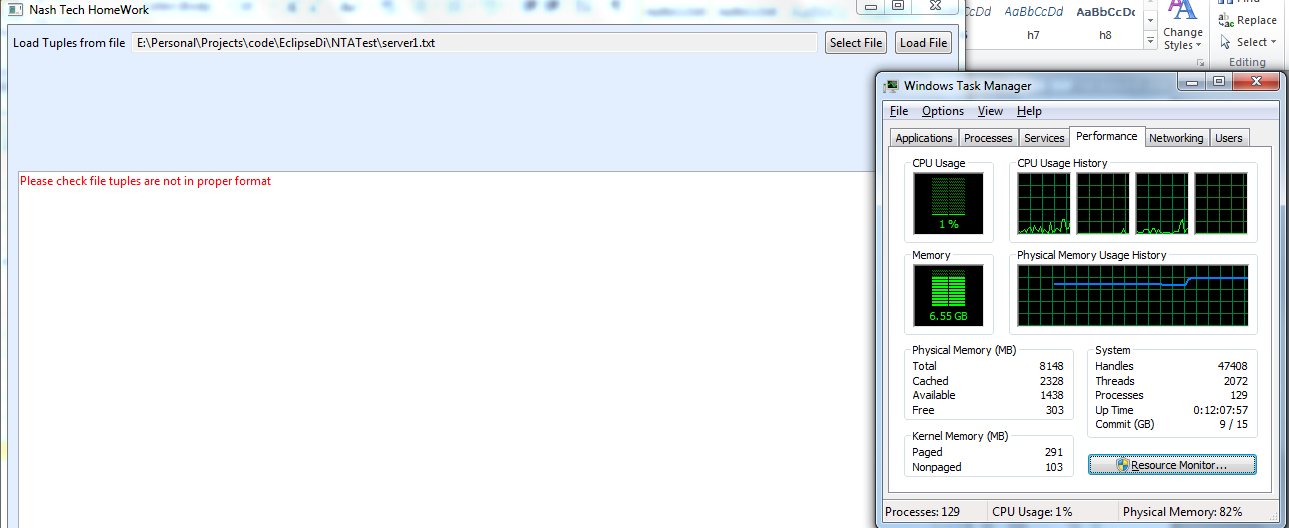


Fig 8. 3