Hanane Nour Moussa

CSC 3374 Advanced Distributed Programming Paradigms

Homework 1: Remote Access Tool

Remote Access Tool

1. RAT Protocol:

The client opens a connection with the server and *informs* it whether it wants to *get running processes in remote system*, *take screenshot of remote system*, or *reboot remote system* using a *header*.

- **Get the list of processes running in the remote system:**

  In this case, the header sent by the client is the following:

  - `processes[line feed]`

  Upon receiving this header, the server searches for the running processes and replies with the following header:

  - `OK [file size][line feed]`
  - followed by the bytes of the processes list.
- **Take screenshot of the remote system:**

  The header sent by the client to take screenshot of remote system is:

  - `screenshot[line feed]`

  The server then responds with:

  - `OK [file size][line feed]`
  - followed by the bytes of the screenshot.
- **Reboot the remote system:**

  To reboot the system, the client sends the following header:

  - `reboot[line feed]`

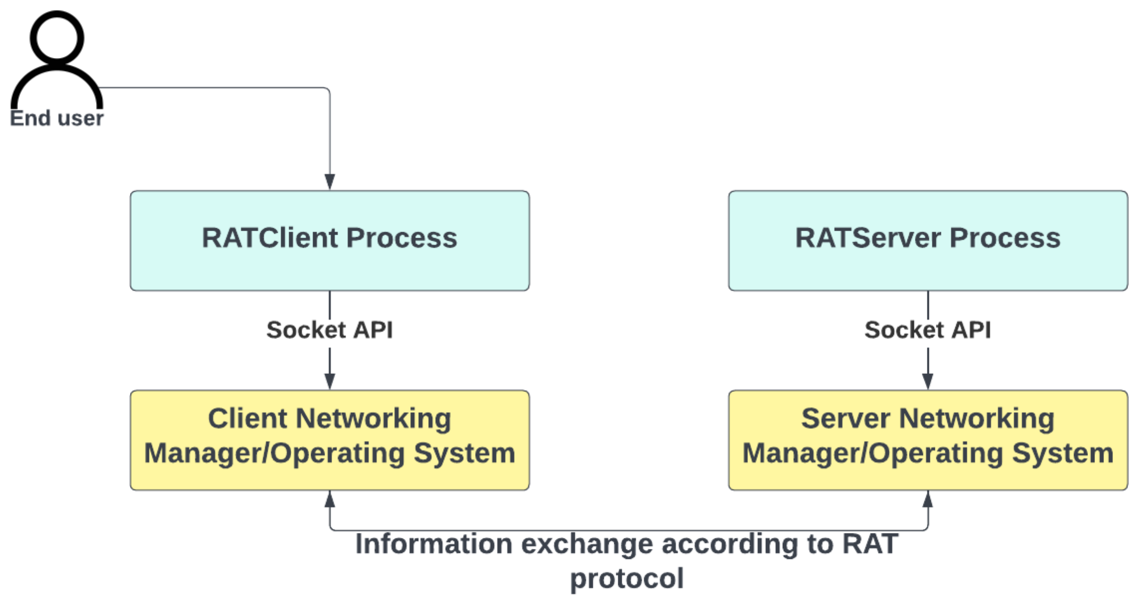
  The server then responds with:

  - `OK[line feed]`

  After which the system reboots after 5 seconds.

2. System Architecture:

The remote access tool was developed following the client server architecture, as shown in the figure below.


End user
RATClient Process
RATServer Process
Socket API
Socket API
Client Networking Manager/Operating System
Server Networking Manager/Operating System
Information exchange according to RAT protocol

RAT client-server architecture

Hence, to make use of the remote access tool, the end user invokes RATClient, which is a process that actively opens a connection with the server. The end user sends a command to the client (whether they want to list running processes, take screenshot, or reboot system) and the client translates these commands to client requests following the RAT protocol. The server accepts requests from the client, processes these requests, and sends back a reply following the RAT protocol.

3. Technology enablers:

The programming language adopted is Java. This choice stems from my familiarity with the language and the wide array of utilities that it offers.

To bind the server to the port number and to establish the client-server connection, I have made use of the `ServerSocket` and `Socket` classes implemented in the `java.net` package.

To retrieve the list of running processes, I have made use of the `ProcessHandle` class implemented in `java.lang`, which offers a wide variety of methods to manipulate processes.

The taking of the screenshot was facilitated by the `Robot` class implemented in `java.awt`.

The `Runtime` class, implemented in `java.lang`, was used to reboot the system.

The implementation was done in the Netbeans IDE using a gradle project.

4. Implementation logic:

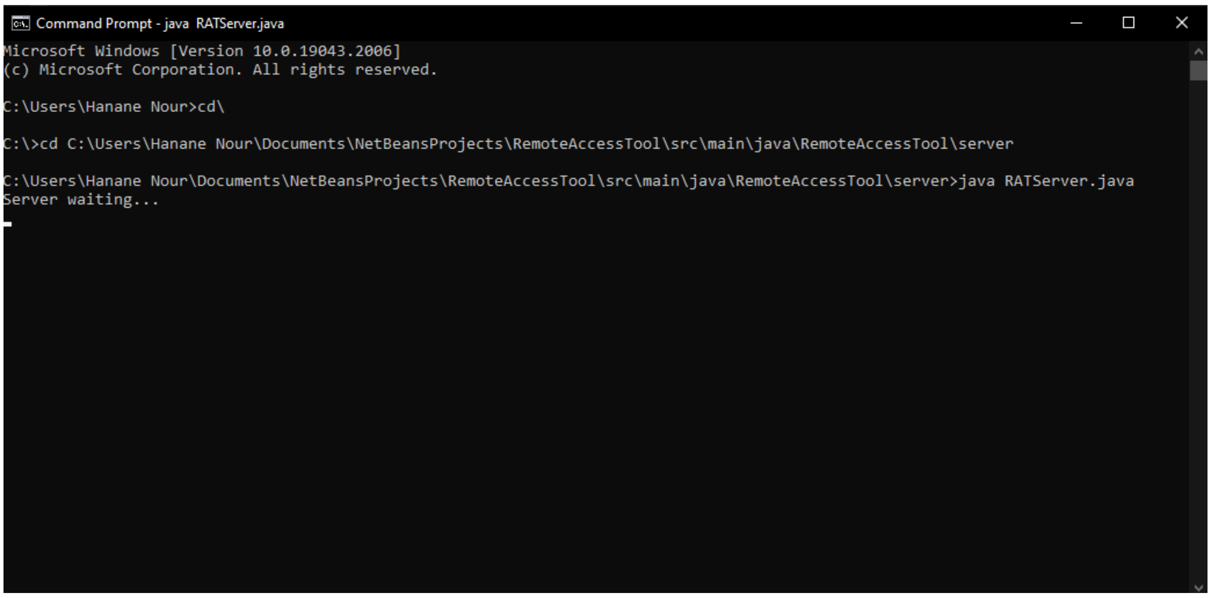
Following is the logic I have adopted to implement the three main functionalities of the system:

- **Get list of running processes:**
  - Client receives "p" argument from end user
  - Client constructs corresponding header
  - Client sends header to server via the output stream

    - Server receives header via the input stream
    - Server gets all processes and filters out the running processes
    - Server constructs a string containing the information list of all processes
    - String is converted to byte array
    - Server response sent to client via output stream
    - Byte array sent to client via data output stream
    - Server response received by client via input stream
    - Client reads in the bytes using data input stream
    - Bytes reconverted to string and printed out to the console
- **Get screenshot:**
    - Client receives “s” argument from end user
    - Client constructs corresponding header
    - Client sends header to server via the output stream
    - Server receives header via the input stream
    - Server takes screenshot of system and stores it in buffered image
    - Buffered image is converted to byte array
    - Server response sent to client via output stream
    - Byte array sent to client via data output stream
    - Server response received by client via input stream
    - Client reads in the bytes using data input stream
    - Client stores the bytes in `C:\Users\[User Name]\Documents\RAT_Screenshot.bmp`
    - Informative message is printed out on the console
- **Reboot:**
    - Client receives “r” argument from end user
    - Client constructs corresponding header
    - Client sends header to server via the output stream
    - Server receives header via the input stream
    - Server response sent to client via output stream
    - Server executes command to reboot the system after 5 seconds
    - Server response received by client via input stream
    - Informative message printed to the console
    - System reboots

5. Demo:

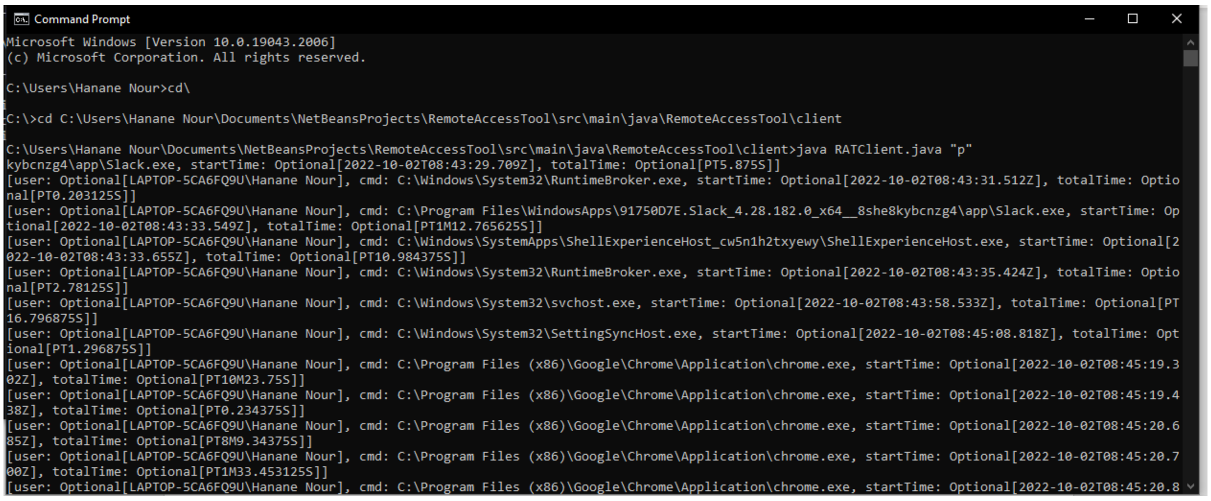
To run the RAT, we first run the server via the command line: java RATServer.java.



```
Microsoft Windows [Version 10.0.19043.2006]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Hanane Nour>cd\

C:\>cd C:\Users\Hanane Nour\Documents\NetBeansProjects\RemoteAccessTool\src\main\java\RemoteAccessTool\server

C:\Users\Hanane Nour\Documents\NetBeansProjects\RemoteAccessTool\src\main\java\RemoteAccessTool\server>java RATServer.java
Server waiting...
```

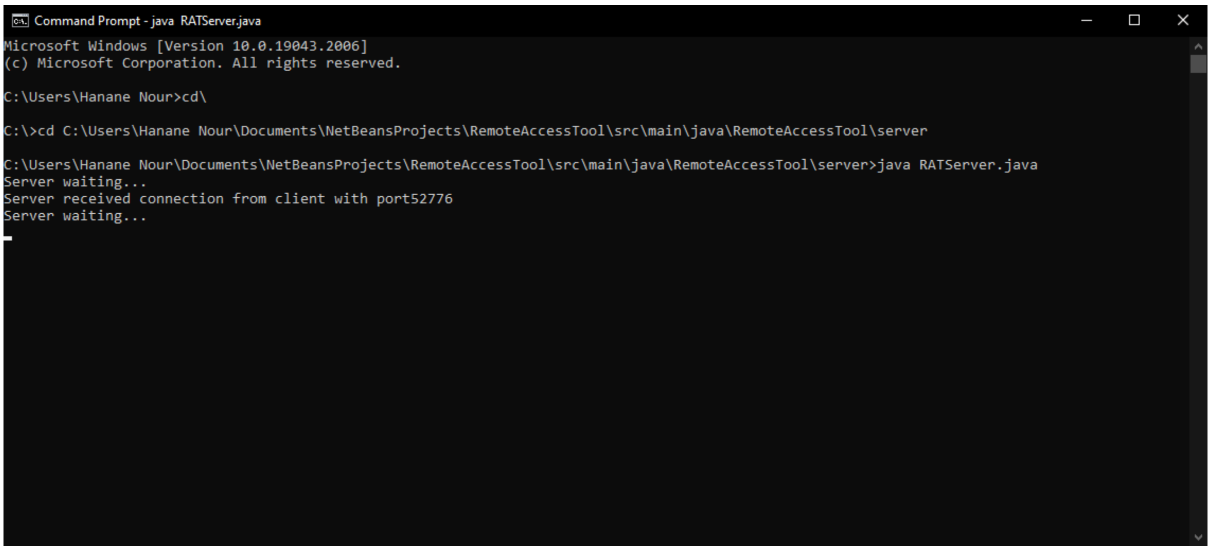
We then invoke the client via a different command prompt.

The running processes are fetched through java RATClient.java “p”:



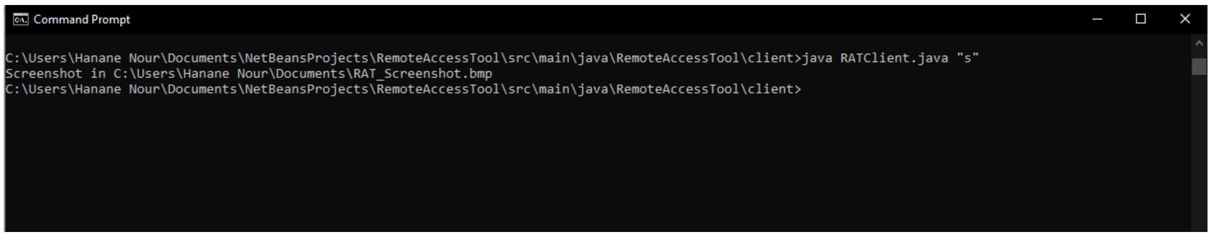
```
Microsoft Windows [Version 10.0.19043.2006]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Hanane Nour>cd\

C:\>cd C:\Users\Hanane Nour\Documents\NetBeansProjects\RemoteAccessTool\src\main\java\RemoteAccessTool\client

C:\Users\Hanane Nour\Documents\NetBeansProjects\RemoteAccessTool\src\main\java\RemoteAccessTool\client>java RATClient.java "p"
kybcnzg4\app\Slack.exe, startTime: Optional[2022-10-02T08:43:29.709Z], totalTime: Optional[PT5.875S]]
[user: Optional[LAPTOP-5CA6FQ9U\Hanane Nour], cmd: C:\Windows\System32\RuntimeBroker.exe, startTime: Optional[2022-10-02T08:43:31.512Z], totalTime: Optio
nal[PT0.203125S]]
[user: Optional[LAPTOP-5CA6FQ9U\Hanane Nour], cmd: C:\Program Files\WindowsApps\91750D7E.Slack_4.28.182.0_x64__8she8kybcnzg4\app\Slack.exe, startTime: Op
tional[2022-10-02T08:43:33.549Z], totalTime: Optional[PT1M12.765625S]]
[user: Optional[LAPTOP-5CA6FQ9U\Hanane Nour], cmd: C:\Windows\SystemApps\ShellExperienceHost_cw5n1h2txyewy\ShellExperienceHost.exe, startTime: Optional[2
022-10-02T08:43:33.655Z], totalTime: Optional[PT10.984375S]]
[user: Optional[LAPTOP-5CA6FQ9U\Hanane Nour], cmd: C:\Windows\System32\RuntimeBroker.exe, startTime: Optional[2022-10-02T08:43:35.424Z], totalTime: Optio
nal[PT2.78125S]]
[user: Optional[LAPTOP-5CA6FQ9U\Hanane Nour], cmd: C:\Windows\System32\svchost.exe, startTime: Optional[2022-10-02T08:43:58.533Z], totalTime: Optional[PT
16.796875S]]
[user: Optional[LAPTOP-5CA6FQ9U\Hanane Nour], cmd: C:\Windows\System32\SettingSyncHost.exe, startTime: Optional[2022-10-02T08:45:08.818Z], totalTime: Opt
ional[PT1.296875S]]
[user: Optional[LAPTOP-5CA6FQ9U\Hanane Nour], cmd: C:\Program Files (x86)\Google\Chrome\Application\chrome.exe, startTime: Optional[2022-10-02T08:45:19.3
02Z], totalTime: Optional[PT10M23.75S]]
[user: Optional[LAPTOP-5CA6FQ9U\Hanane Nour], cmd: C:\Program Files (x86)\Google\Chrome\Application\chrome.exe, startTime: Optional[2022-10-02T08:45:19.4
38Z], totalTime: Optional[PT0.234375S]]
[user: Optional[LAPTOP-5CA6FQ9U\Hanane Nour], cmd: C:\Program Files (x86)\Google\Chrome\Application\chrome.exe, startTime: Optional[2022-10-02T08:45:20.6
85Z], totalTime: Optional[PT8M9.34375S]]
[user: Optional[LAPTOP-5CA6FQ9U\Hanane Nour], cmd: C:\Program Files (x86)\Google\Chrome\Application\chrome.exe, startTime: Optional[2022-10-02T08:45:20.7
00Z], totalTime: Optional[PT1M33.453125S]]
[user: Optional[LAPTOP-5CA6FQ9U\Hanane Nour], cmd: C:\Program Files (x86)\Google\Chrome\Application\chrome.exe, startTime: Optional[2022-10-02T08:45:20.8
```

On the server side, we can also see the port number of the client.



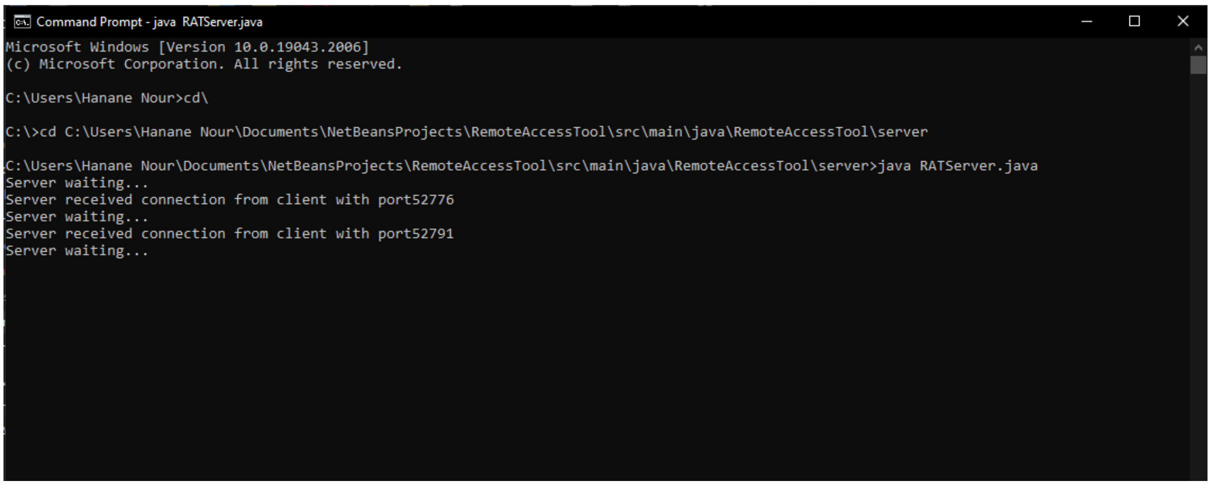
```
Microsoft Windows [Version 10.0.19043.2006]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Hanane Nour>cd\

C:\>cd C:\Users\Hanane Nour\Documents\NetBeansProjects\RemoteAccessTool\src\main\java\RemoteAccessTool\server

C:\Users\Hanane Nour\Documents\NetBeansProjects\RemoteAccessTool\src\main\java\RemoteAccessTool\server>java RATServer.java
Server waiting...
Server received connection from client with port52776
Server waiting...
```

The screenshot is take through java RATClient.java “s”:



```
C:\Users\Hanane Nour\Documents\NetBeansProjects\RemoteAccessTool\src\main\java\RemoteAccessTool\client>java RATClient.java "s"
Screenshot in C:\Users\Hanane Nour\Documents\RAT_Screenshot.bmp
C:\Users\Hanane Nour\Documents\NetBeansProjects\RemoteAccessTool\src\main\java\RemoteAccessTool\client>
```

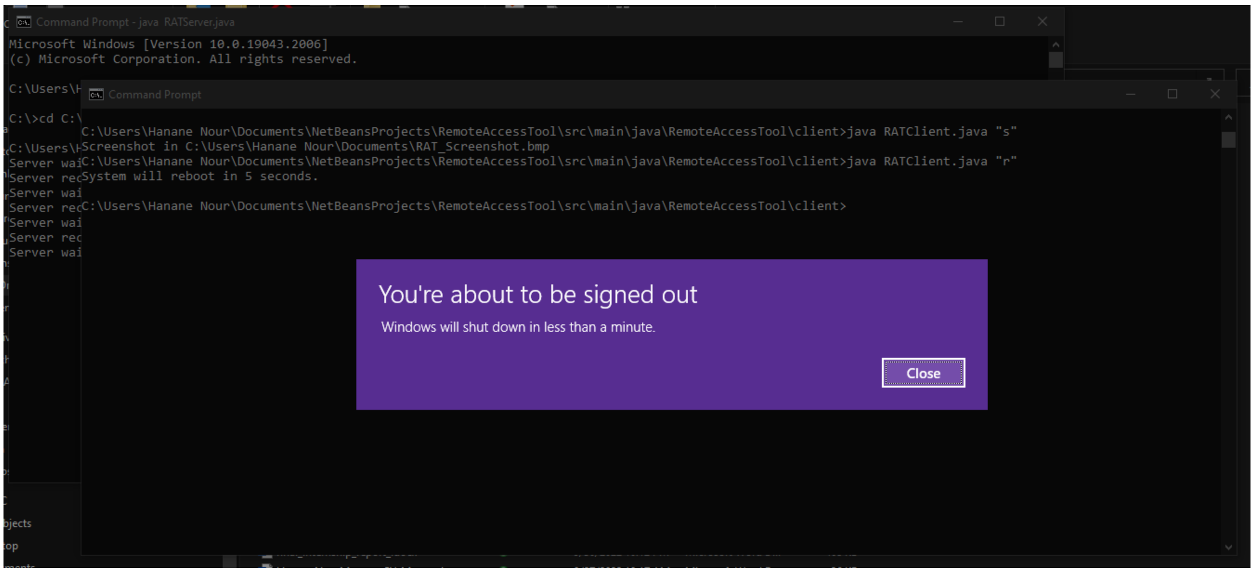
As expected, the port number of the client changes across different requests.



```
Microsoft Windows [Version 10.0.19043.2006]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Hanane Nour>cd\

C:\>cd C:\Users\Hanane Nour\Documents\NetBeansProjects\RemoteAccessTool\src\main\java\RemoteAccessTool\server

C:\Users\Hanane Nour\Documents\NetBeansProjects\RemoteAccessTool\src\main\java\RemoteAccessTool\server>java RATServer.java
Server waiting...
Server received connection from client with port52776
Server waiting...
Server received connection from client with port52791
Server waiting...
```

The screenshot can then be found in the path specified.

To reboot the system, we execute the command java RATClient.java “r”:



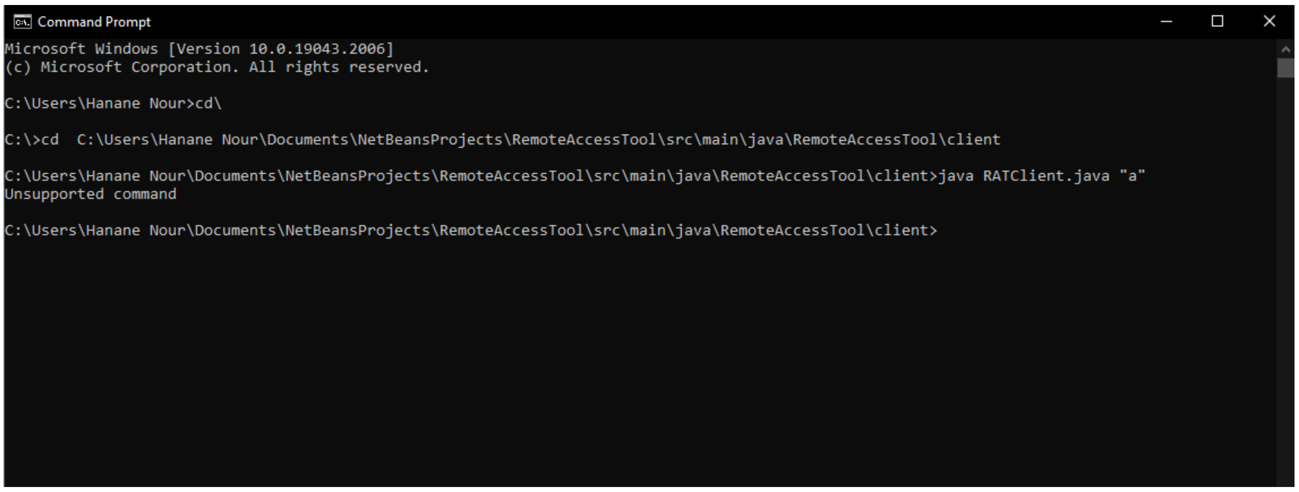
```
C:\Users\Hanane Nour\Documents\NetBeansProjects\RemoteAccessTool\src\main\java\RemoteAccessTool\client>java RATClient.java "s"
Screenshot in C:\Users\Hanane Nour\Documents\RAT_Screenshot.bmp
C:\Users\Hanane Nour\Documents\NetBeansProjects\RemoteAccessTool\src\main\java\RemoteAccessTool\client>java RATClient.java "r"
System will reboot in 5 seconds.

C:\Users\Hanane Nour\Documents\NetBeansProjects\RemoteAccessTool\src\main\java\RemoteAccessTool\client>
```

You're about to be signed out

Windows will shut down in less than a minute.

After which the system reboots.

If an argument other then “p”, “s”, or “r” is given, the message “unsupported command” is printed to the console:



Command Prompt

```
Microsoft Windows [Version 10.0.19043.2006]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Hanane Nour>cd\

C:\>cd  C:\Users\Hanane Nour\Documents\NetBeansProjects\RemoteAccessTool\src\main\java\RemoteAccessTool\client

C:\Users\Hanane Nour\Documents\NetBeansProjects\RemoteAccessTool\src\main\java\RemoteAccessTool\client>java RATClient.java "a"
Unsupported command

C:\Users\Hanane Nour\Documents\NetBeansProjects\RemoteAccessTool\src\main\java\RemoteAccessTool\client>
```