Project Documentation
CSE 310
Wei Tang
Stephen Egert

Overview

As for now, we have implemented both stage one and stage two. Major implementation includes type, done, help, put, get, del, browse and exit. Please notice that the implementation of Stage 2, we have dropped all type fields in command.

Major Function:

type [t]
    return the port number of the DNS server of t type and establish TCP connection between client and DNS server. Return not found if no DNS with given type.
done
    close current TCP connection with DNS server and establish connection with manager application
help
    print help message
put [name] [IP addr]
    put command will put a DNS record with name field, ip address and the type assigned to this specific DNS server.
get [name]
    get command will return the IP address of the given name, return not found
if   no record with given name is founded.
del [name]
    del command will delete the record in the DNS server with the given name. return not found if no record with given name in database
browse
    browse all the record in the database within the current DNS server.
exit
    close TCP connection. Client application exits.

User documentation

Our implementation using runtime.exec() method to create multiple server with distinct type and with in a single DNS server, we use java thread class to handle concurrent client requests. Java thread class is also used in manager to serve multi clients at the same time. We never really test the limit for my project. In my test cases it supports 5 server at the same time.

To compile manager application (Server side):

javac multiserver.java
javac manager.java

(need to compile both java files)

To run manager application:

java manager.java

To compile and run client side code

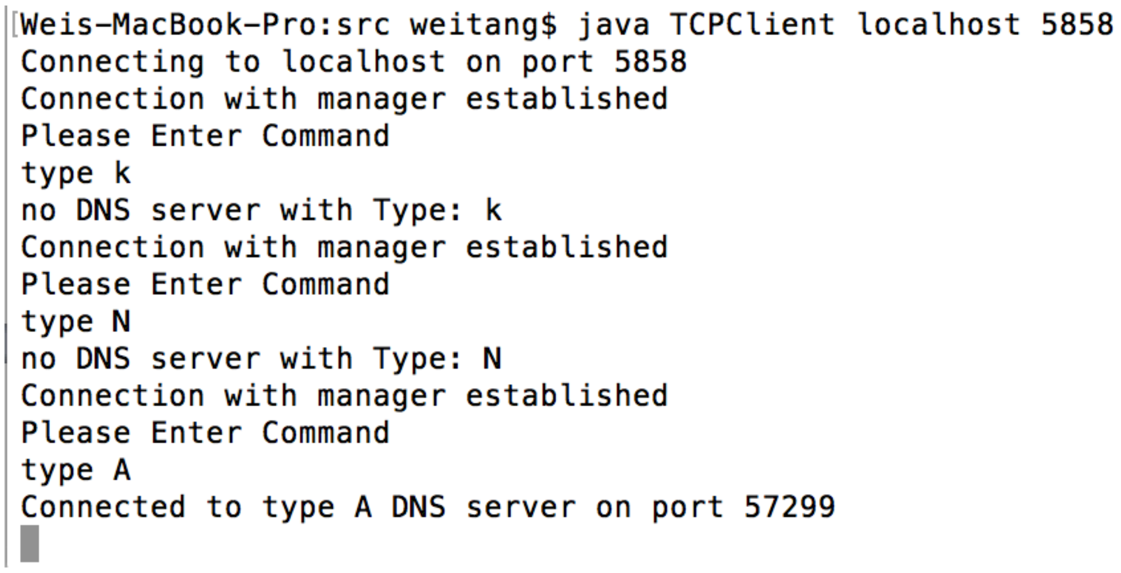
javac TCPClient.java
java TCPClient <hostname> 5858

(Our implementation of manager application is on port 5858)

Once the connection between manager and client established client will promote user to input command, at this stage only command help, type[type] and exit can be used (because now the client is connected to manager not a DNS server)

<u>Syntax and parameters for my program:</u>

Example for connect to type N DNS server (with error cases):

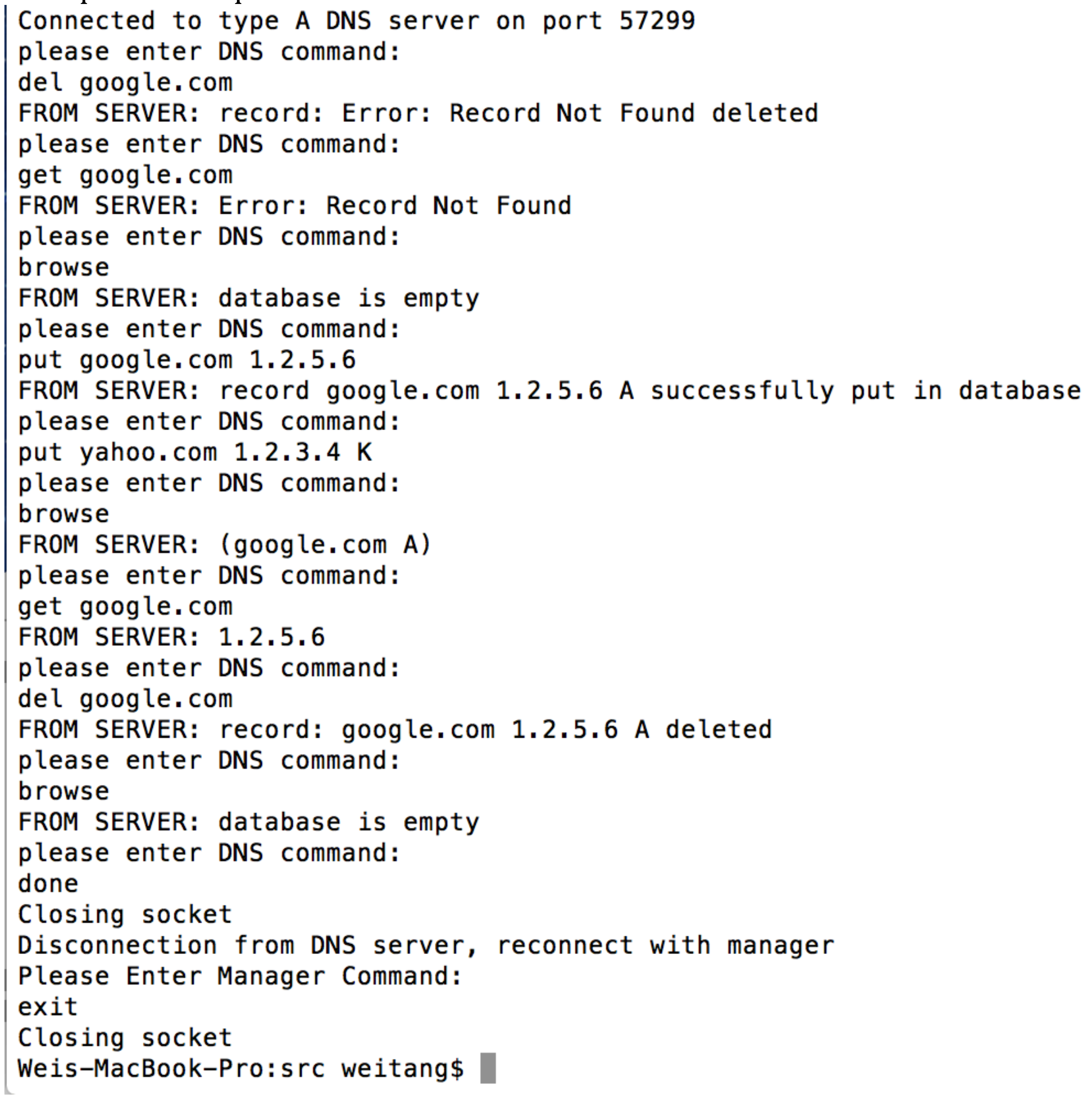


```
Weis-MacBook-Pro:src weitang$ java TCPClient localhost 5858
Connecting to localhost on port 5858
Connection with manager established
Please Enter Command
type k
no DNS server with Type: k
Connection with manager established
Please Enter Command
type N
no DNS server with Type: N
Connection with manager established
Please Enter Command
type A
Connected to type A DNS server on port 57299
```

Possible error message here would be “no DNS server with type found” or “unknown manager command” for wrong command

Once connection between a DNS and client is established then all other function is now available to user (browse, get, del, done and help)

Example for a sequence of command:



```
Connected to type A DNS server on port 57299
please enter DNS command:
del google.com
FROM SERVER: record: Error: Record Not Found deleted
please enter DNS command:
get google.com
FROM SERVER: Error: Record Not Found
please enter DNS command:
browse
FROM SERVER: database is empty
please enter DNS command:
put google.com 1.2.5.6
FROM SERVER: record google.com 1.2.5.6 A successfully put in database
please enter DNS command:
put yahoo.com 1.2.3.4 K
please enter DNS command:
browse
FROM SERVER: (google.com A)
please enter DNS command:
get google.com
FROM SERVER: 1.2.5.6
please enter DNS command:
del google.com
FROM SERVER: record: google.com 1.2.5.6 A deleted
please enter DNS command:
browse
FROM SERVER: database is empty
please enter DNS command:
done
Closing socket
Disconnection from DNS server, reconnect with manager
Please Enter Manager Command:
exit
Closing socket
Weis-MacBook-Pro:src weitang$
```

possible error message:

“Database empty” for empty database
“record not found” for no matching record in database for both get and del
notice that in the sequence put yahoo.com 1.2.3.4 K. this command does not put the record to the data base because we drop the type field in the command which make it an illegal command.

Once user enters done, connection between DNS and client will be closed and establish new connection to manager for using other DNS servers.

System documentation

<u>Main Java Classes:</u>

Manager class:

Manager application classes that parse the manager.io and launch corresponding DNS servers(in different processes), record their type and port number in an ArrayList and listen to client connection. Once connection is established pass the connection to ManagerServiceThread Class

ManagerServiceThread Class:

managerServiceThread class extends thread class. The thread will handle client requests.

Multiserver class:

Multiserver class is a DNS server with a specific type and a random assigned port number. It will pipe the port number using getLocalPort() method and pass it to the Manager Class for it to store service record. It will listen to the socket and pass socket connection to a ClientServiceThread class once it establish TCP connection with a client.

ClientServiceThread class:

ClientServiceThread class extends thread class. It will listen to clients commands and output the correct respond

<u>Communication Protocol:</u>

The communication between clients and manager or server is through a pipe.
Client can send message using writeUTF()
Server or manager can send message using readUTF()
All responses are print in terminal.

<u>Data Structor:</u>

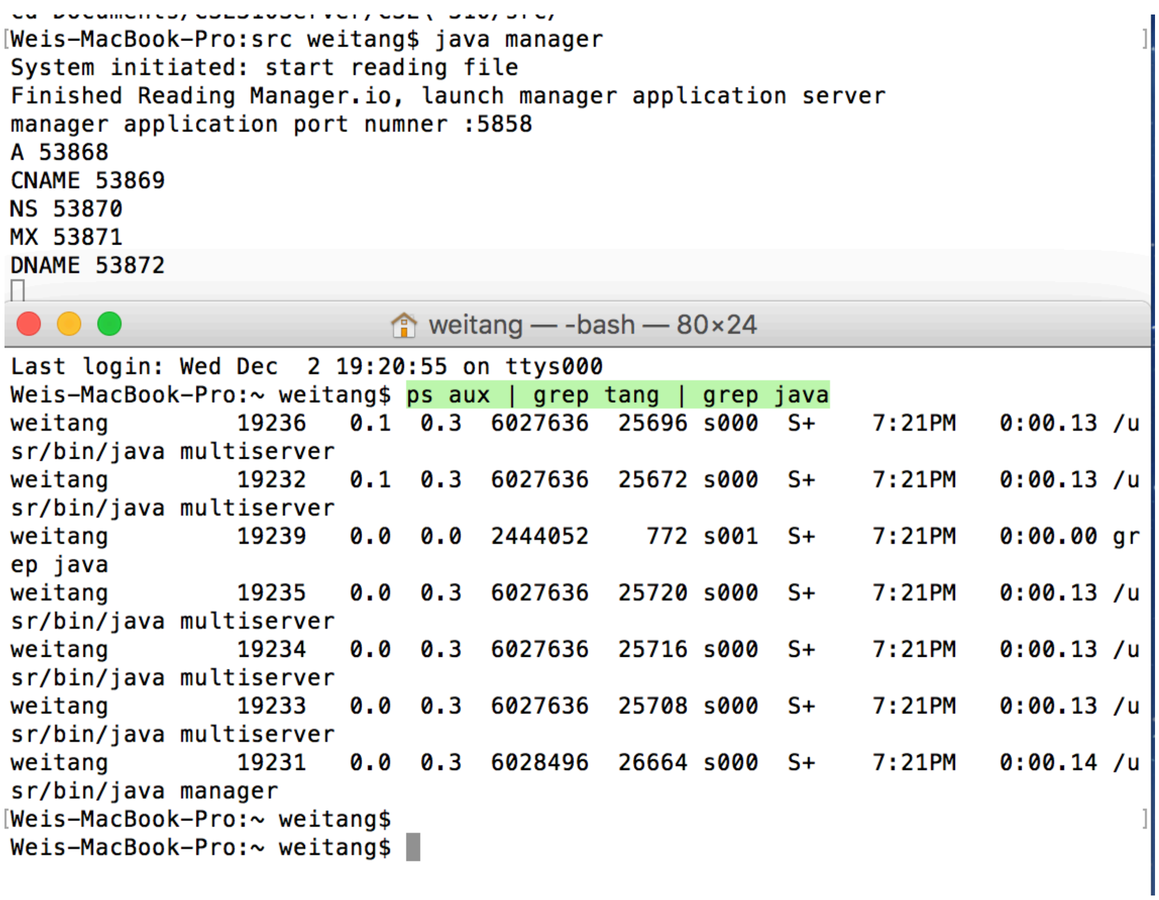
We use different txt file for different DNS server. if the DNS server is type N then the database file for it would have the name of “N.txt”
The data record format : [name] [IP_address] [type]

<u>Error Conditions:</u>

Error case most involve unknown command from client and data record not exist.

Test Documentation:

Test case 1: DNS server start up in manager class (create process in manager):
Reason: multi process worth 30 percent of the project
Precondition: all file compiled, manager.in file exists
Step: 1. Run "java manager"
2. ps aux | grep tang | grep java
Expect result: one manager process with several multiserver processes (depend on manager.in)
Actual result: 1 manager process with 5 multiserver processes



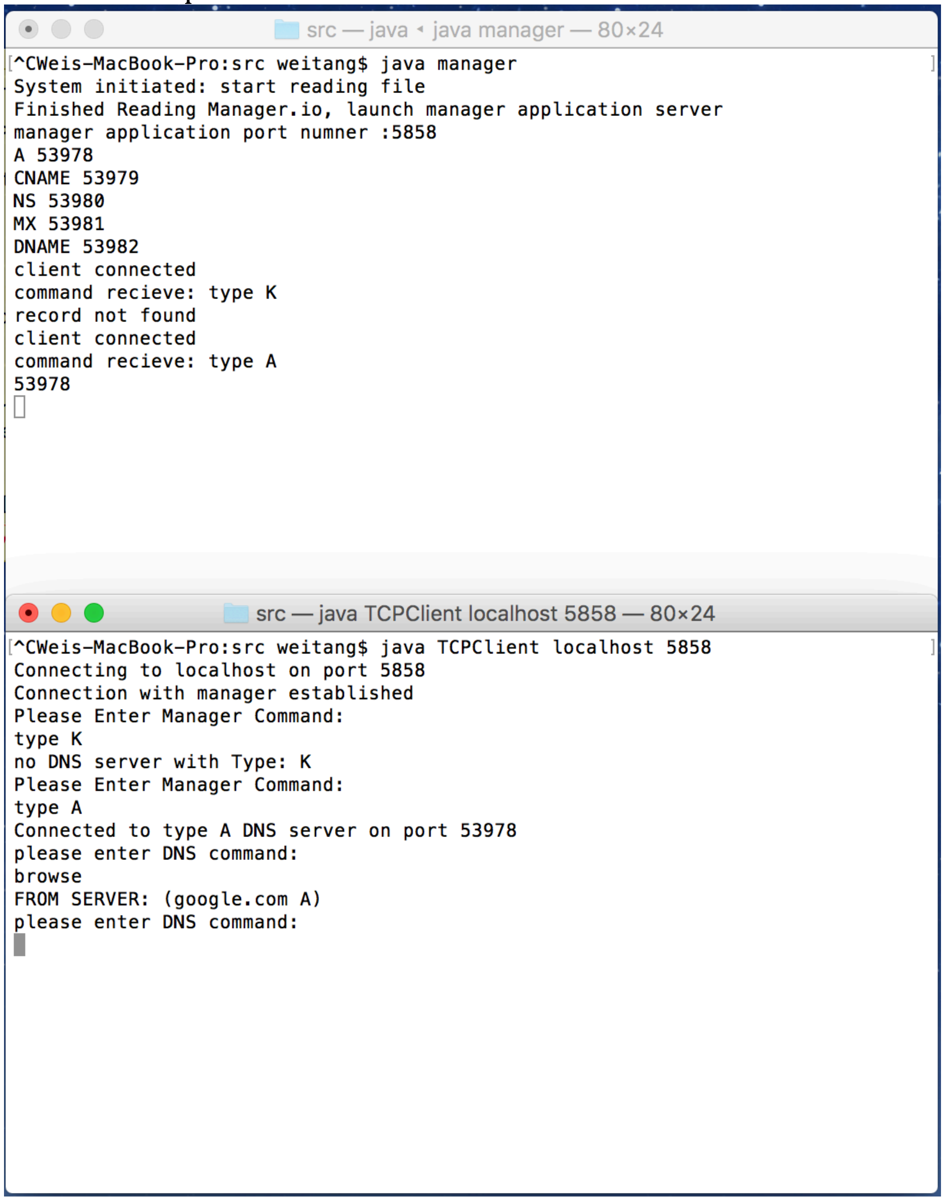
```
Weis-MacBook-Pro:src weitang$ java manager
System initiated: start reading file
Finished Reading Manager.io, launch manager application server
manager application port numner :5858
A 53868
CNAME 53869
NS 53870
MX 53871
DNAME 53872
```

weitang — -bash — 80×24

```
Last login: Wed Dec  2 19:20:55 on ttys000
Weis-MacBook-Pro:~ weitang$ ps aux | grep tang | grep java
weitang          19236   0.1  0.3  6027636  25696 s000  S+    7:21PM   0:00.13 /u
sr/bin/java multiserver
weitang          19232   0.1  0.3  6027636  25672 s000  S+    7:21PM   0:00.13 /u
sr/bin/java multiserver
weitang          19239   0.0  0.0  2444052    772 s001  S+    7:21PM   0:00.00 gr
ep java
weitang          19235   0.0  0.3  6027636  25720 s000  S+    7:21PM   0:00.13 /u
sr/bin/java multiserver
weitang          19234   0.0  0.3  6027636  25716 s000  S+    7:21PM   0:00.13 /u
sr/bin/java multiserver
weitang          19233   0.0  0.3  6027636  25708 s000  S+    7:21PM   0:00.13 /u
sr/bin/java multiserver
weitang          19231   0.0  0.3  6028496  26664 s000  S+    7:21PM   0:00.14 /u
sr/bin/java manager
Weis-MacBook-Pro:~ weitang$
Weis-MacBook-Pro:~ weitang$
```

Test Case 2: TYPE command
Reason: functionality for manager and implementation of automatically connection to DNS server after client's inquiry for a specific DNS server

Precondition: all file complied. Manager application launched
Step: 1. "java TCPClient <host address> 5858"
2. type K
3.type A
Expected result:
after 2: error message should returned
after 3: client should connect to type A DNS server and be able to use the DNS server
Actual result:
Work perfect see screenshots.


src — java ◂ java manager — 80×24

```
^CWeis-MacBook-Pro:src weitang$ java manager
System initiated: start reading file
Finished Reading Manager.io, launch manager application server
manager application port numner :5858
A 53978
CNAME 53979
NS 53980
MX 53981
DNAME 53982
client connected
command recieve: type K
record not found
client connected
command recieve: type A
53978
```

src — java TCPClient localhost 5858 — 80×24

```
^CWeis-MacBook-Pro:src weitang$ java TCPClient localhost 5858
Connecting to localhost on port 5858
Connection with manager established
Please Enter Manager Command:
type K
no DNS server with Type: K
Please Enter Manager Command:
type A
Connected to type A DNS server on port 53978
please enter DNS command:
browse
FROM SERVER: (google.com A)
please enter DNS command:
```

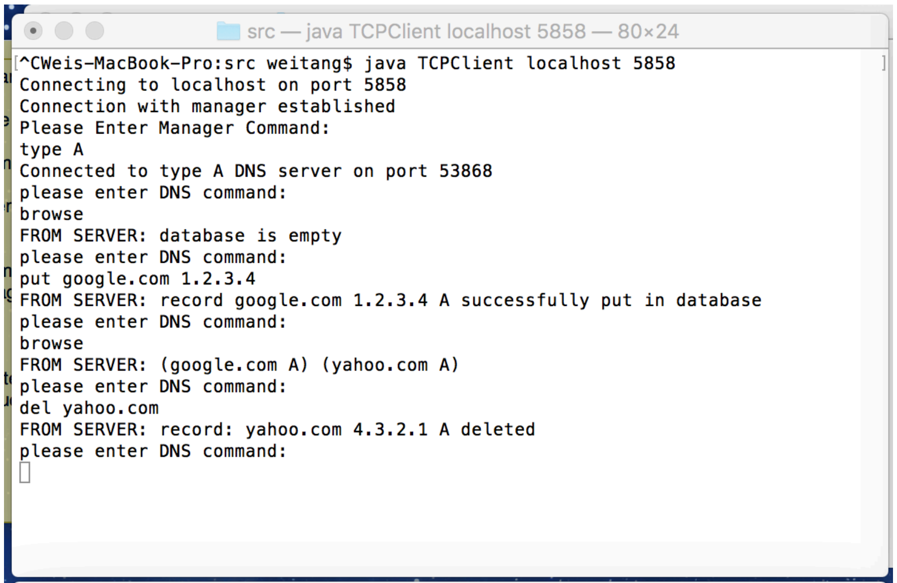
Test case 3: a sequence of put, get, del and browse command with multi clients
Reason: major function test and concurrent client test. It also test concurrency of manager.
Precondition: database empty, 2 clients connected to a DNS server
Step:
1. "browse" to make sure data base is empty
2. "put google.com 1.2.3.4" on client 1
3. "put yahoo.com 4.3.2.1" on client 2
4. "browse" on either client
5. "del yahoo.com" on either client
6. "get google.com" on either client
7. "get yahoo.com" on either client
8. "browse" on either client

Expect result:

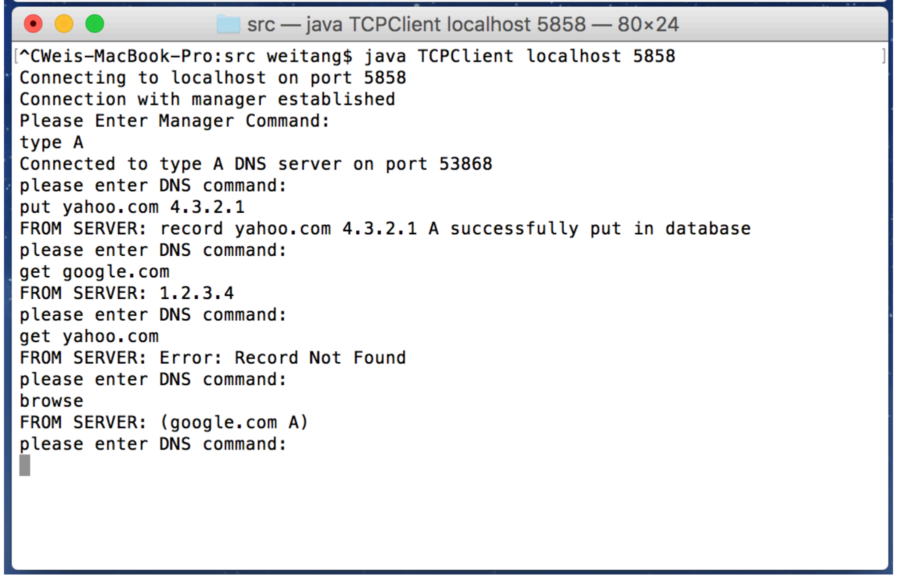
After step 4. Terminal should show 2 record with correct information
After step 6. Terminal should show google.com's IP address
After step 7. Error case should return
After step 8. Only 1 record should shown

Actual result:

(next page screenshot)
work perfect

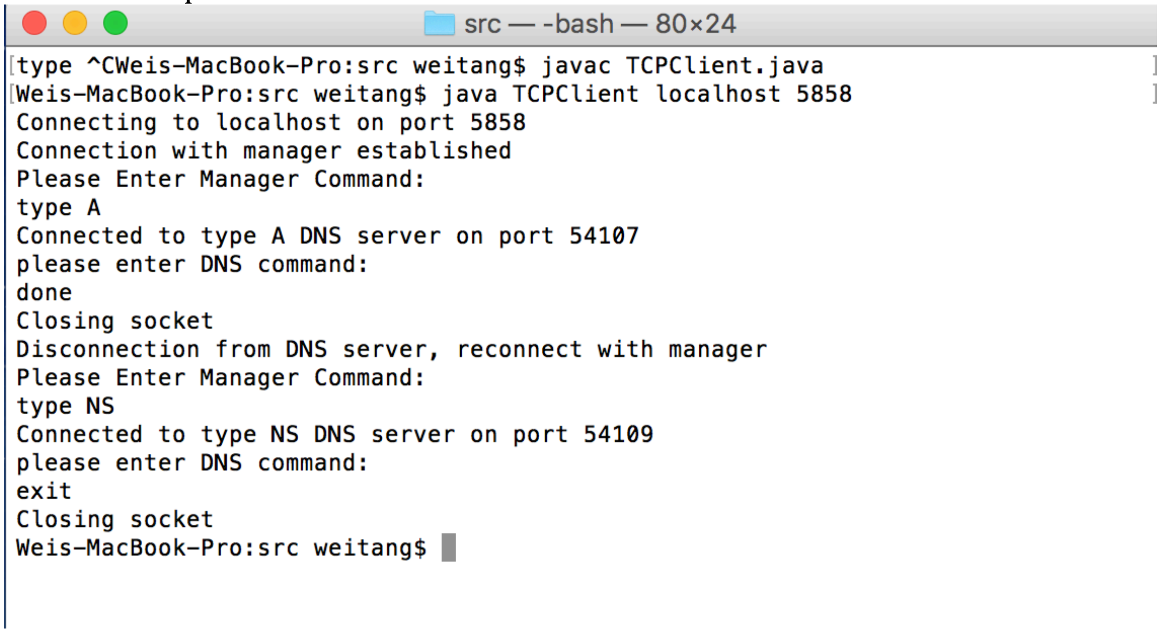


```
^CWeis-MacBook-Pro:src weitang$ java TCPClient localhost 5858
Connecting to localhost on port 5858
Connection with manager established
Please Enter Manager Command:
type A
Connected to type A DNS server on port 53868
please enter DNS command:
browse
FROM SERVER: database is empty
please enter DNS command:
put google.com 1.2.3.4
FROM SERVER: record google.com 1.2.3.4 A successfully put in database
please enter DNS command:
browse
FROM SERVER: (google.com A) (yahoo.com A)
please enter DNS command:
del yahoo.com
FROM SERVER: record: yahoo.com 4.3.2.1 A deleted
please enter DNS command:
```



```
^CWeis-MacBook-Pro:src weitang$ java TCPClient localhost 5858
Connecting to localhost on port 5858
Connection with manager established
Please Enter Manager Command:
type A
Connected to type A DNS server on port 53868
please enter DNS command:
put yahoo.com 4.3.2.1
FROM SERVER: record yahoo.com 4.3.2.1 A successfully put in database
please enter DNS command:
get google.com
FROM SERVER: 1.2.3.4
please enter DNS command:
get yahoo.com
FROM SERVER: Error: Record Not Found
please enter DNS command:
browse
FROM SERVER: (google.com A)
please enter DNS command:
```

Test Case 4: done command and exit command
Reason: done should disconnect client from DNS server and reconnect it to manager
Exit should exit client application
Precondition: client is connect to manager.
Step: 1. “type A”
2. “done”
3. “type NS”
4. “exit”
Expected result:
After 2. Client should reconnect to manager
After 5. Client app should terminated.
Actual result:
Work perfect



```
[type ^CWeis-MacBook-Pro:src weitang$ javac TCPClient.java                  ]
[Weis-MacBook-Pro:src weitang$ java TCPClient localhost 5858                ]
Connecting to localhost on port 5858
Connection with manager established
Please Enter Manager Command:
type A
Connected to type A DNS server on port 54107
please enter DNS command:
done
Closing socket
Disconnection from DNS server, reconnect with manager
Please Enter Manager Command:
type NS
Connected to type NS DNS server on port 54109
please enter DNS command:
exit
Closing socket
Weis-MacBook-Pro:src weitang$
```