HW45 Project Report

Halim Acosta

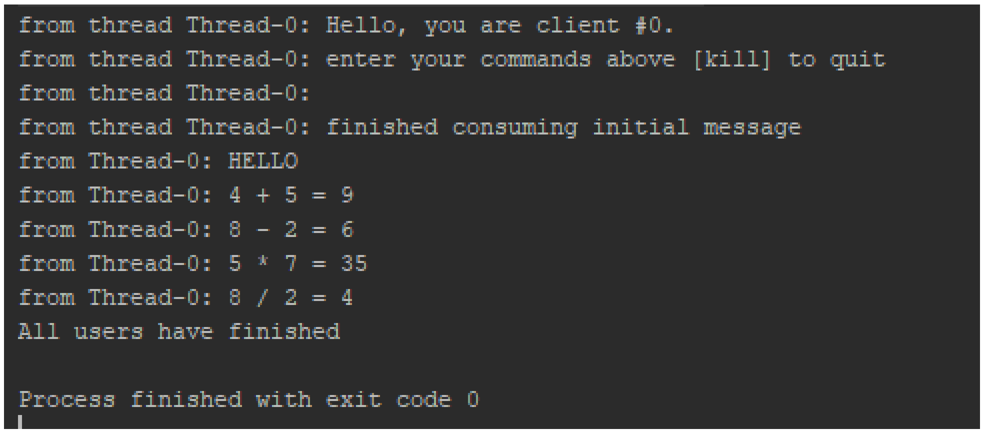
7/27/2018

Class Design and Testing:

For this project I have designed three classes that work together to form the server and a fourth class for testing the server without using the GUI components that were given as the initial startup code.

The main class that aggregates all the data and handles specifics is the ThreadPool class. This class was designed to maintain a pool of threads that waits on a job queue that services user requests. Each of the main threads also have access to an auxiliary thread pool in case the thread that's servicing a connection needs more resource use. The ThreadPool class also has a timer object that repeatedly queries the queue for how many jobs are pending as well as querying how many of the threads in the pool are actively servicing a job. The timer object will then perform the necessary actions to regulate the thread usage whether that be to increase the number of threads in the pool or decrease it. The queue is controlled by a Monitor class that controls the access to and from the queue. The queue holds a CustomJob class that carries its input and output streams, the clients id number as well as the service the thread plans to provide. The class comes with different methods for creating Runnables for the different types of commands that server can do. It also comes with an error msg runnable in case there is some kind of error and a toString() method for printing out the job along with its time stamp. The final class that was designed was the ServerTester class which is used to simulate several connections on the server querying different commands. The class supports sending single or multiple commands to the server.

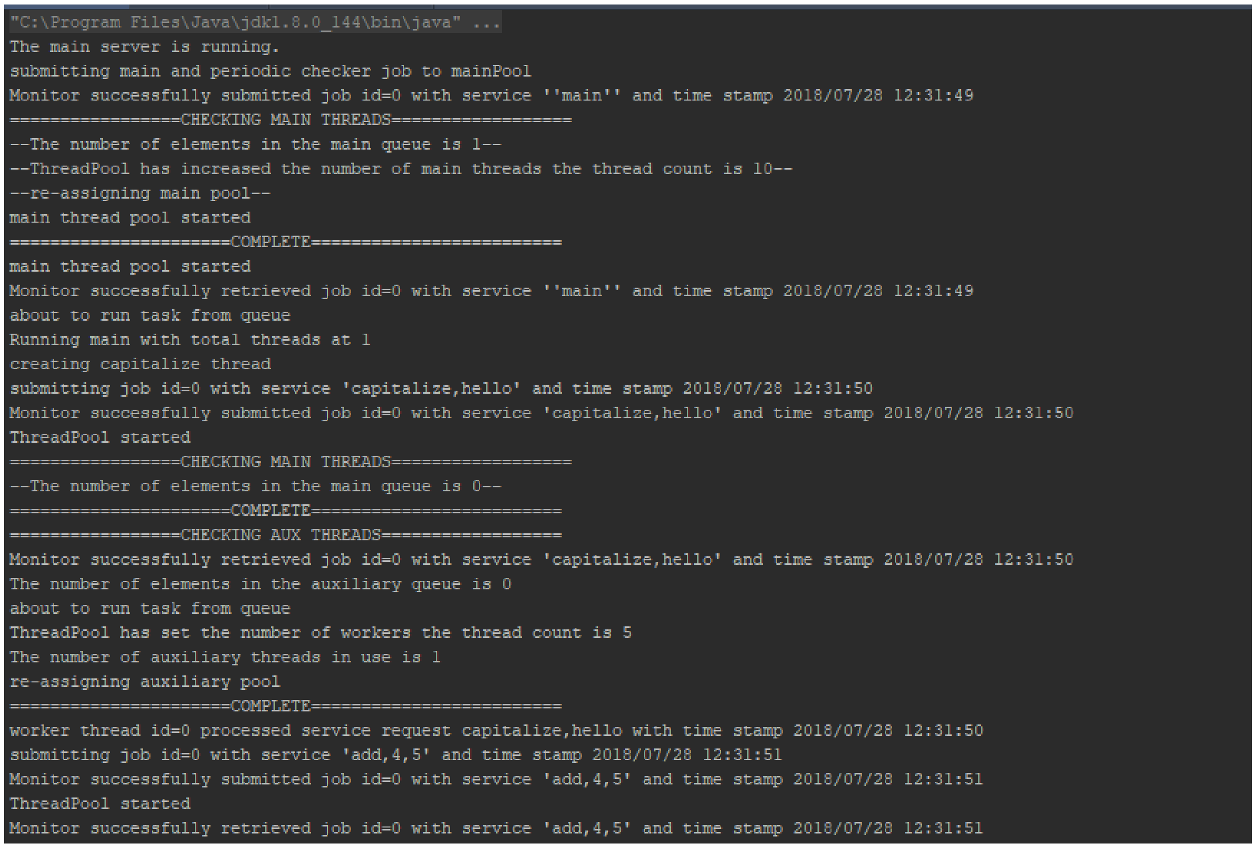
Below is sample output from the tester showing a single user performing each of the supported commands and then killing the server



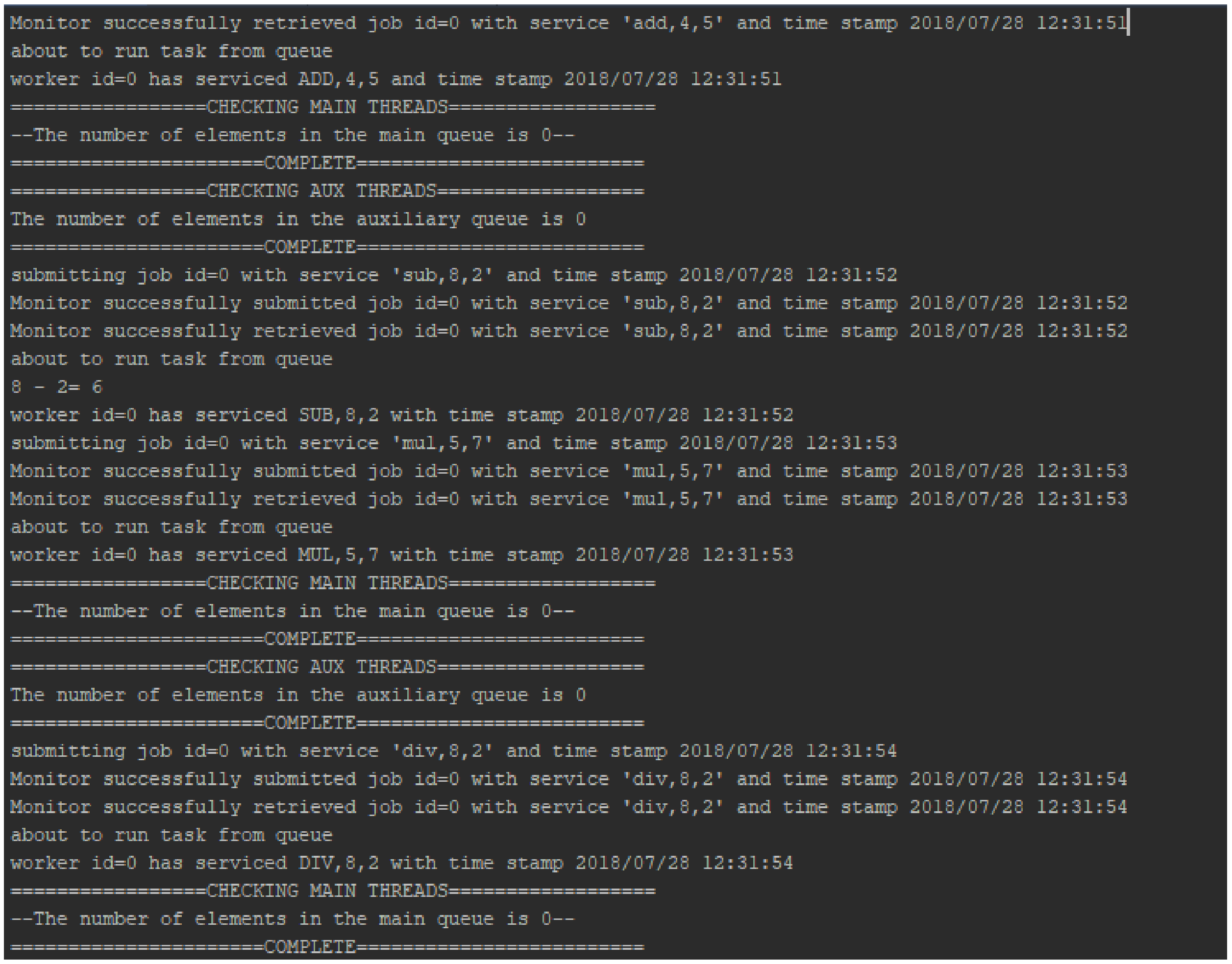
```
from thread Thread-0: Hello, you are client #0.
from thread Thread-0: enter your commands above [kill] to quit
from thread Thread-0:
from thread Thread-0: finished consuming initial message
from Thread-0: HELLO
from Thread-0: 4 + 5 = 9
from Thread-0: 8 - 2 = 6
from Thread-0: 5 * 7 = 35
from Thread-0: 8 / 2 = 4
All users have finished

Process finished with exit code 0
```

This user connects to the server and consumes the greeting message and then starts by using the capitalize command and then using each of the subsequent arithmetic commands before it sends the 'KILL' command server and exits

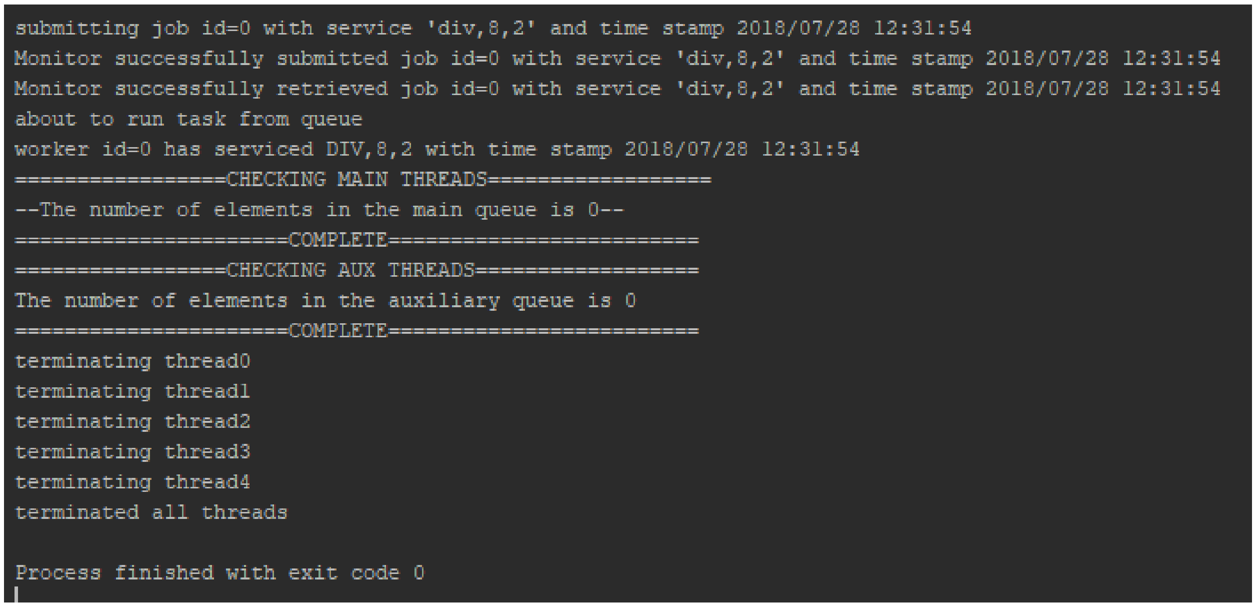
Below is the complete output from the server for this interaction



```
"C:\Program Files\Java\jdk1.8.0_144\bin\java" ...
The main server is running.
submitting main and periodic checker job to mainPool
Monitor successfully submitted job id=0 with service ''main'' and time stamp 2018/07/28 12:31:49
=================CHECKING MAIN THREADS==================
--The number of elements in the main queue is 1--
--ThreadPool has increased the number of main threads the thread count is 10--
--re-assigning main pool--
main thread pool started
======================COMPLETE=========================
main thread pool started
Monitor successfully retrieved job id=0 with service ''main'' and time stamp 2018/07/28 12:31:49
about to run task from queue
Running main with total threads at 1
creating capitalize thread
submitting job id=0 with service 'capitalize,hello' and time stamp 2018/07/28 12:31:50
Monitor successfully submitted job id=0 with service 'capitalize,hello' and time stamp 2018/07/28 12:31:50
ThreadPool started
=================CHECKING MAIN THREADS==================
--The number of elements in the main queue is 0--
======================COMPLETE=========================
=================CHECKING AUX THREADS==================
Monitor successfully retrieved job id=0 with service 'capitalize,hello' and time stamp 2018/07/28 12:31:50
The number of elements in the auxiliary queue is 0
about to run task from queue
ThreadPool has set the number of workers the thread count is 5
The number of auxiliary threads in use is 1
re-assigning auxiliary pool
======================COMPLETE=========================
worker thread id=0 processed service request capitalize,hello with time stamp 2018/07/28 12:31:50
submitting job id=0 with service 'add,4,5' and time stamp 2018/07/28 12:31:51
Monitor successfully submitted job id=0 with service 'add,4,5' and time stamp 2018/07/28 12:31:51
ThreadPool started
Monitor successfully retrieved job id=0 with service 'add,4,5' and time stamp 2018/07/28 12:31:51
```

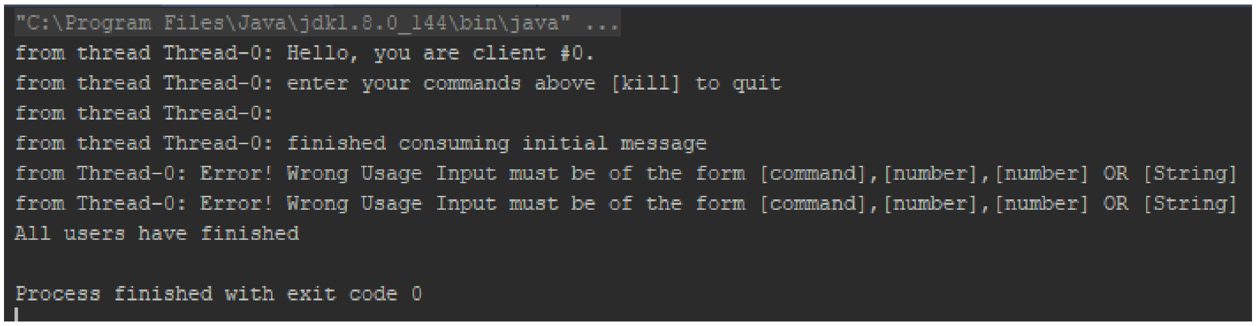


```
Monitor successfully retrieved job id=0 with service 'add,4,5' and time stamp 2018/07/28 12:31:51
about to run task from queue
worker id=0 has serviced ADD,4,5 and time stamp 2018/07/28 12:31:51
=================CHECKING MAIN THREADS==================
--The number of elements in the main queue is 0--
======================COMPLETE=========================
=================CHECKING AUX THREADS==================
The number of elements in the auxiliary queue is 0
======================COMPLETE=========================
submitting job id=0 with service 'sub,8,2' and time stamp 2018/07/28 12:31:52
Monitor successfully submitted job id=0 with service 'sub,8,2' and time stamp 2018/07/28 12:31:52
Monitor successfully retrieved job id=0 with service 'sub,8,2' and time stamp 2018/07/28 12:31:52
about to run task from queue
8 - 2= 6
worker id=0 has serviced SUB,8,2 with time stamp 2018/07/28 12:31:52
submitting job id=0 with service 'mul,5,7' and time stamp 2018/07/28 12:31:53
Monitor successfully submitted job id=0 with service 'mul,5,7' and time stamp 2018/07/28 12:31:53
Monitor successfully retrieved job id=0 with service 'mul,5,7' and time stamp 2018/07/28 12:31:53
about to run task from queue
worker id=0 has serviced MUL,5,7 with time stamp 2018/07/28 12:31:53
=================CHECKING MAIN THREADS==================
--The number of elements in the main queue is 0--
======================COMPLETE=========================
=================CHECKING AUX THREADS==================
The number of elements in the auxiliary queue is 0
======================COMPLETE=========================
submitting job id=0 with service 'div,8,2' and time stamp 2018/07/28 12:31:54
Monitor successfully submitted job id=0 with service 'div,8,2' and time stamp 2018/07/28 12:31:54
Monitor successfully retrieved job id=0 with service 'div,8,2' and time stamp 2018/07/28 12:31:54
about to run task from queue
worker id=0 has serviced DIV,8,2 with time stamp 2018/07/28 12:31:54
=================CHECKING MAIN THREADS==================
--The number of elements in the main queue is 0--
======================COMPLETE=========================
```



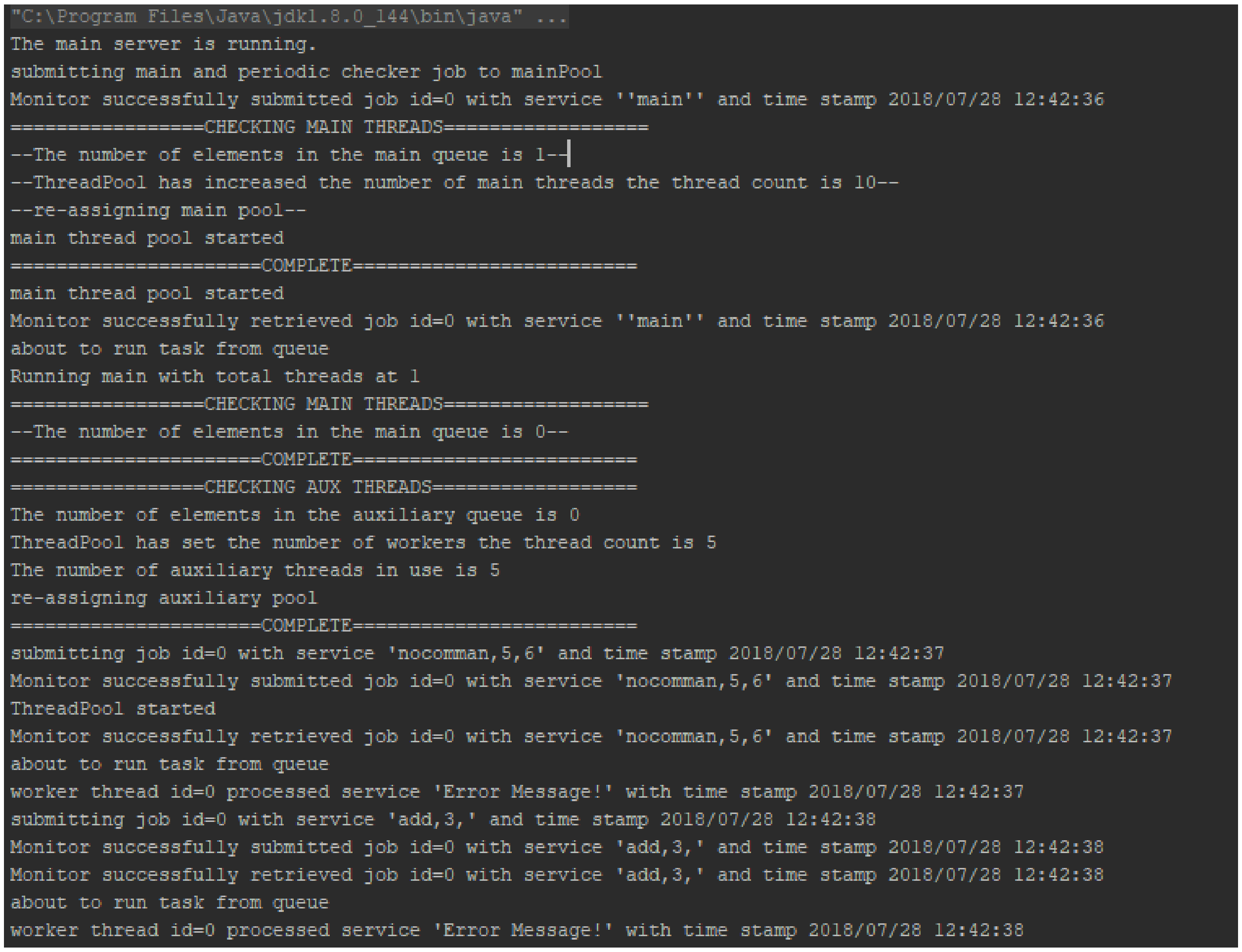
```
submitting job id=0 with service 'div,8,2' and time stamp 2018/07/28 12:31:54
Monitor successfully submitted job id=0 with service 'div,8,2' and time stamp 2018/07/28 12:31:54
Monitor successfully retrieved job id=0 with service 'div,8,2' and time stamp 2018/07/28 12:31:54
about to run task from queue
worker id=0 has serviced DIV,8,2 with time stamp 2018/07/28 12:31:54
=================CHECKING MAIN THREADS==================
--The number of elements in the main queue is 0--
======================COMPLETE=========================
=================CHECKING AUX THREADS==================
The number of elements in the auxiliary queue is 0
======================COMPLETE=========================
terminating thread0
terminating thread1
terminating thread2
terminating thread3
terminating thread4
terminated all threads

Process finished with exit code 0
```

Next will be the output from a single user sending erroneous ouput

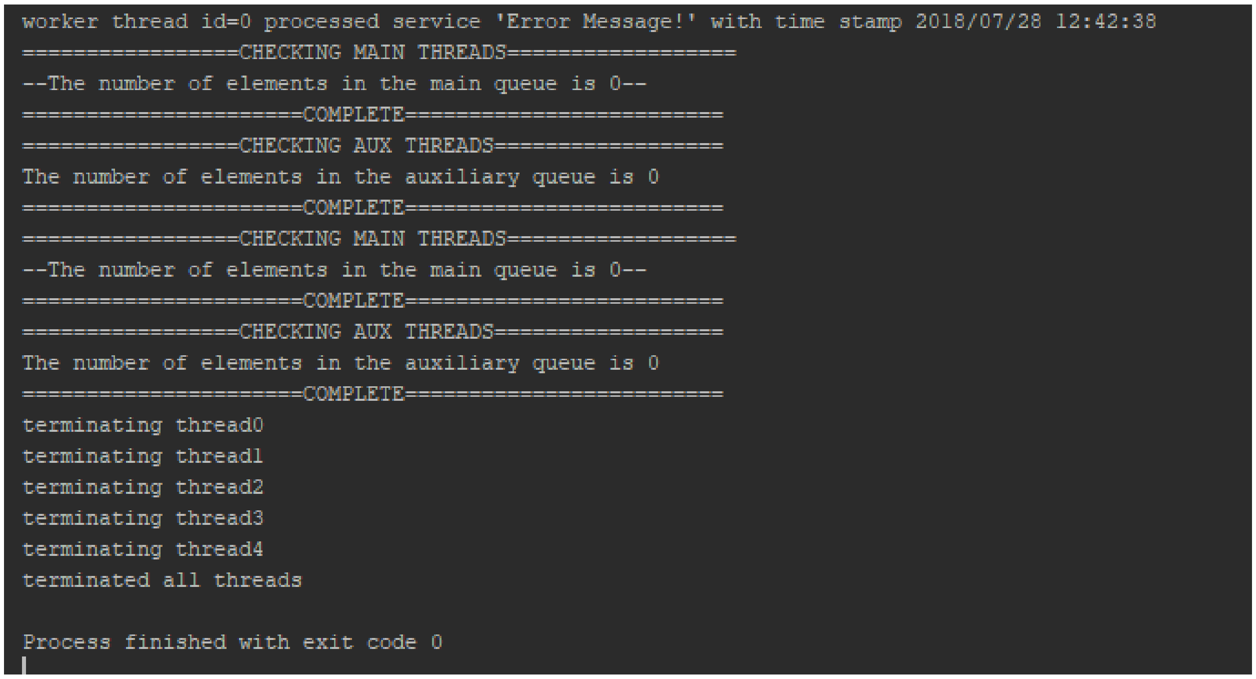


```
"C:\Program Files\Java\jdk1.8.0_144\bin\java" ...
from thread Thread-0: Hello, you are client #0.
from thread Thread-0: enter your commands above [kill] to quit
from thread Thread-0:
from thread Thread-0: finished consuming initial message
from Thread-0: Error! Wrong Usage Input must be of the form [command],[number],[number] OR [String]
from Thread-0: Error! Wrong Usage Input must be of the form [command],[number],[number] OR [String]
All users have finished

Process finished with exit code 0
```

And below is the output from the server with this interaction

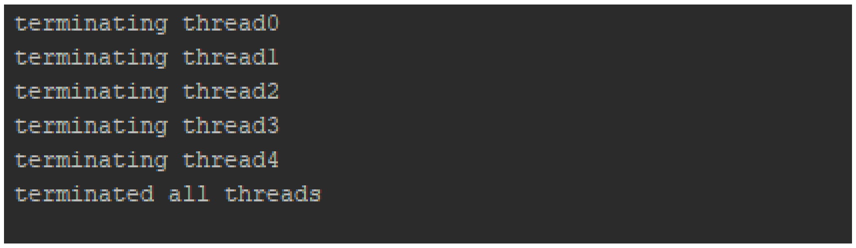


```
"C:\Program Files\Java\jdk1.8.0_144\bin\java" ...
The main server is running.
submitting main and periodic checker job to mainPool
Monitor successfully submitted job id=0 with service ''main'' and time stamp 2018/07/28 12:42:36
=================CHECKING MAIN THREADS==================
--The number of elements in the main queue is 1--
--ThreadPool has increased the number of main threads the thread count is 10--
--re-assigning main pool--
main thread pool started
======================COMPLETE========================
main thread pool started
Monitor successfully retrieved job id=0 with service ''main'' and time stamp 2018/07/28 12:42:36
about to run task from queue
Running main with total threads at 1
=================CHECKING MAIN THREADS==================
--The number of elements in the main queue is 0--
======================COMPLETE========================
=================CHECKING AUX THREADS=================
The number of elements in the auxiliary queue is 0
ThreadPool has set the number of workers the thread count is 5
The number of auxiliary threads in use is 5
re-assigning auxiliary pool
======================COMPLETE========================
submitting job id=0 with service 'nocomman,5,6' and time stamp 2018/07/28 12:42:37
Monitor successfully submitted job id=0 with service 'nocomman,5,6' and time stamp 2018/07/28 12:42:37
ThreadPool started
Monitor successfully retrieved job id=0 with service 'nocomman,5,6' and time stamp 2018/07/28 12:42:37
about to run task from queue
worker thread id=0 processed service 'Error Message!' with time stamp 2018/07/28 12:42:37
submitting job id=0 with service 'add,3,' and time stamp 2018/07/28 12:42:38
Monitor successfully submitted job id=0 with service 'add,3,' and time stamp 2018/07/28 12:42:38
Monitor successfully retrieved job id=0 with service 'add,3,' and time stamp 2018/07/28 12:42:38
about to run task from queue
worker thread id=0 processed service 'Error Message!' with time stamp 2018/07/28 12:42:38
```



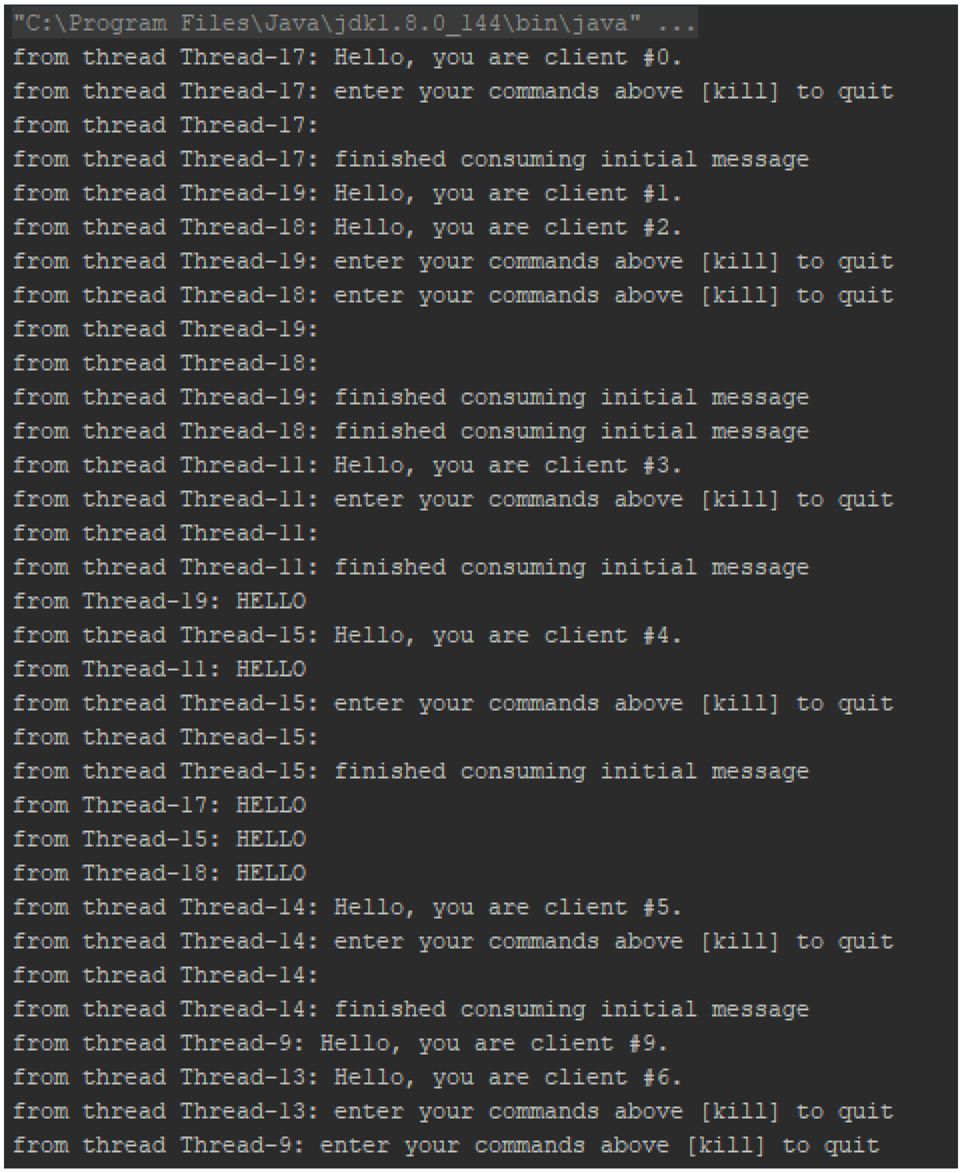
```
worker thread id=0 processed service 'Error Message!' with time stamp 2018/07/28 12:42:38
=================CHECKING MAIN THREADS==================
--The number of elements in the main queue is 0--
======================COMPLETE=========================
=================CHECKING AUX THREADS==================
The number of elements in the auxiliary queue is 0
======================COMPLETE=========================
=================CHECKING MAIN THREADS==================
--The number of elements in the main queue is 0--
======================COMPLETE=========================
=================CHECKING AUX THREADS==================
The number of elements in the auxiliary queue is 0
======================COMPLETE=========================
terminating thread0
terminating thread1
terminating thread2
terminating thread3
terminating thread4
terminated all threads

Process finished with exit code 0
```

When a kill command is sent to the server the server shutdown process is as below

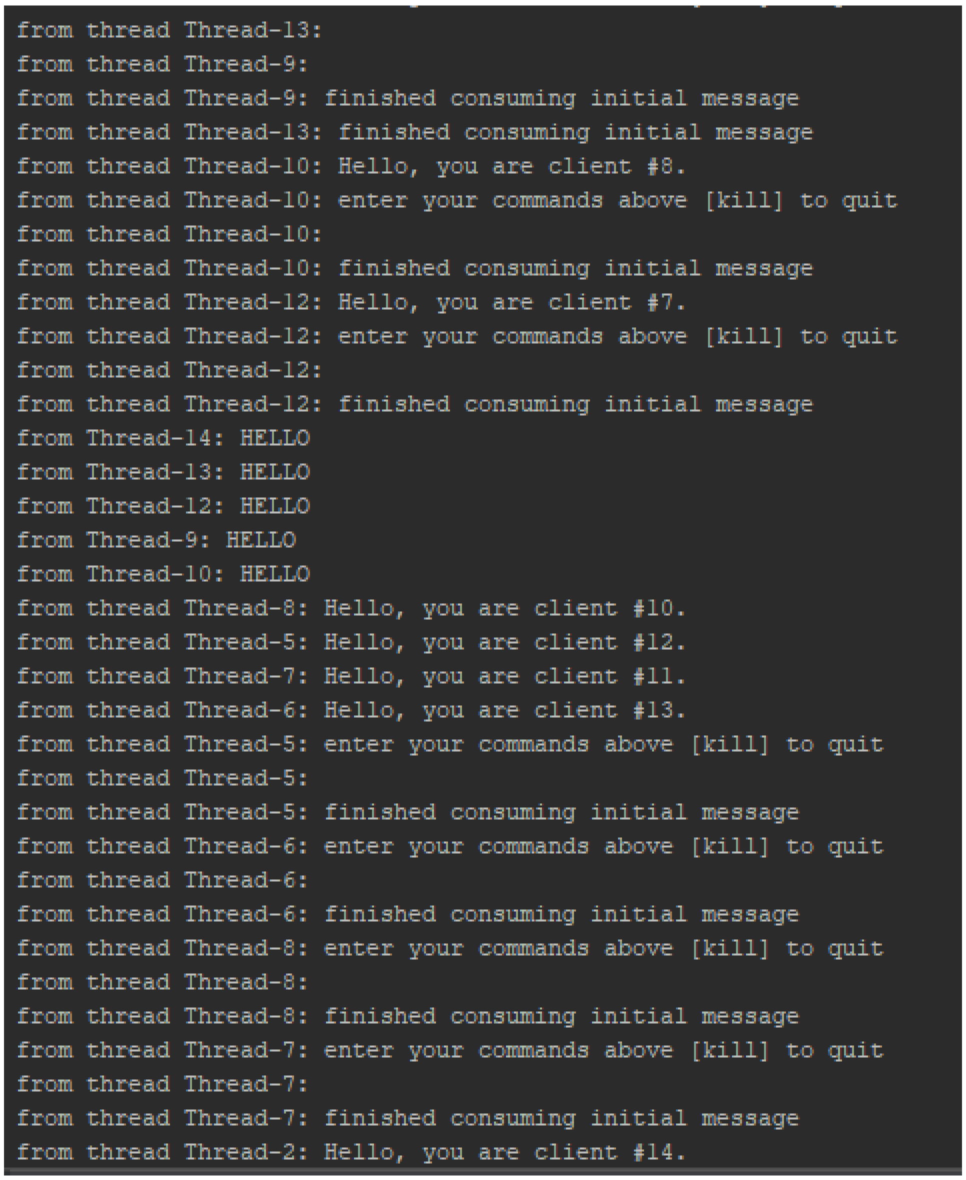


```
terminating thread0
terminating thread1
terminating thread2
terminating thread3
terminating thread4
terminated all threads
```

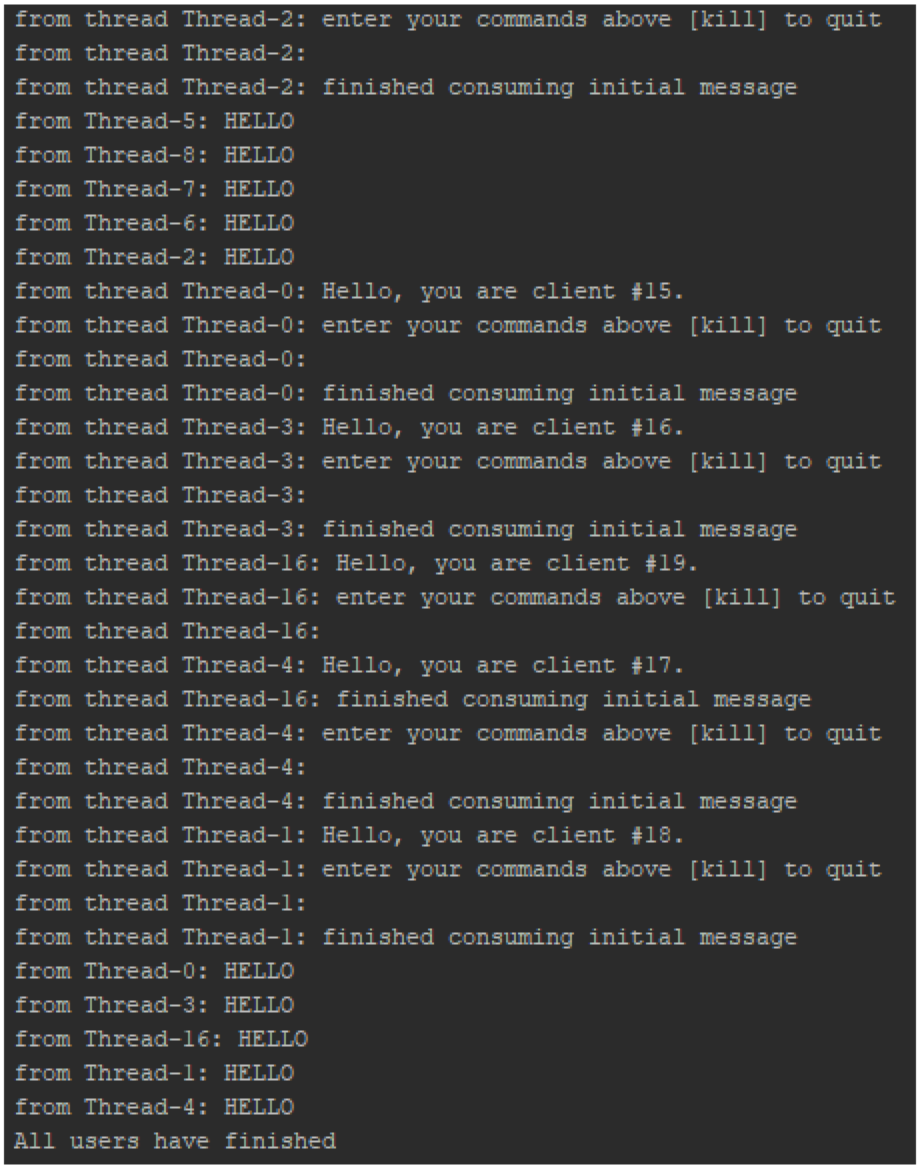
To test what happens when the server is queried by many ‘users’ at once the tester class creates 20 connections to the server and then sends a string to be capitalized. The Tester output is below



```
"C:\Program Files\Java\jdk1.8.0_144\bin\java" ...
from thread Thread-17: Hello, you are client #0.
from thread Thread-17: enter your commands above [kill] to quit
from thread Thread-17:
from thread Thread-17: finished consuming initial message
from thread Thread-19: Hello, you are client #1.
from thread Thread-18: Hello, you are client #2.
from thread Thread-19: enter your commands above [kill] to quit
from thread Thread-18: enter your commands above [kill] to quit
from thread Thread-19:
from thread Thread-18:
from thread Thread-19: finished consuming initial message
from thread Thread-18: finished consuming initial message
from thread Thread-11: Hello, you are client #3.
from thread Thread-11: enter your commands above [kill] to quit
from thread Thread-11:
from thread Thread-11: finished consuming initial message
from Thread-19: HELLO
from thread Thread-15: Hello, you are client #4.
from Thread-11: HELLO
from thread Thread-15: enter your commands above [kill] to quit
from thread Thread-15:
from thread Thread-15: finished consuming initial message
from Thread-17: HELLO
from Thread-15: HELLO
from Thread-18: HELLO
from thread Thread-14: Hello, you are client #5.
from thread Thread-14: enter your commands above [kill] to quit
from thread Thread-14:
from thread Thread-14: finished consuming initial message
from thread Thread-9: Hello, you are client #9.
from thread Thread-13: Hello, you are client #6.
from thread Thread-13: enter your commands above [kill] to quit
from thread Thread-9: enter your commands above [kill] to quit
```

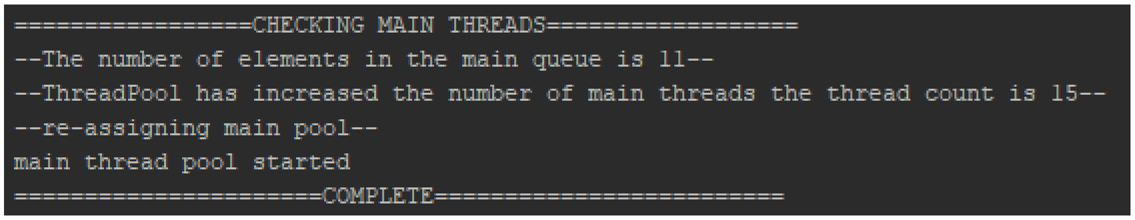


```
from thread Thread-13:
from thread Thread-9:
from thread Thread-9: finished consuming initial message
from thread Thread-13: finished consuming initial message
from thread Thread-10: Hello, you are client #8.
from thread Thread-10: enter your commands above [kill] to quit
from thread Thread-10:
from thread Thread-10: finished consuming initial message
from thread Thread-12: Hello, you are client #7.
from thread Thread-12: enter your commands above [kill] to quit
from thread Thread-12:
from thread Thread-12: finished consuming initial message
from Thread-14: HELLO
from Thread-13: HELLO
from Thread-12: HELLO
from Thread-9: HELLO
from Thread-10: HELLO
from thread Thread-8: Hello, you are client #10.
from thread Thread-5: Hello, you are client #12.
from thread Thread-7: Hello, you are client #11.
from thread Thread-6: Hello, you are client #13.
from thread Thread-5: enter your commands above [kill] to quit
from thread Thread-5:
from thread Thread-5: finished consuming initial message
from thread Thread-6: enter your commands above [kill] to quit
from thread Thread-6:
from thread Thread-6: finished consuming initial message
from thread Thread-8: enter your commands above [kill] to quit
from thread Thread-8:
from thread Thread-8: finished consuming initial message
from thread Thread-7: enter your commands above [kill] to quit
from thread Thread-7:
from thread Thread-7: finished consuming initial message
from thread Thread-2: Hello, you are client #14.
```



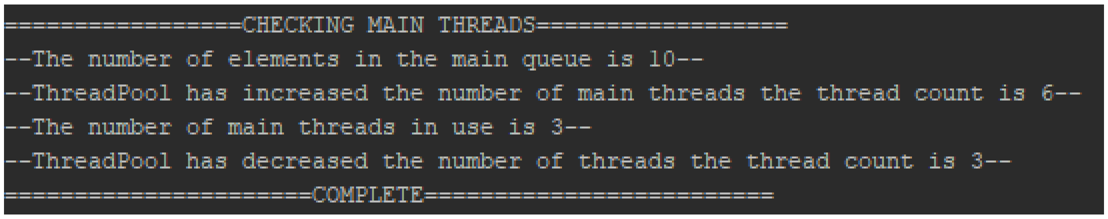
```
from thread Thread-2: enter your commands above [kill] to quit
from thread Thread-2:
from thread Thread-2: finished consuming initial message
from Thread-5: HELLO
from Thread-8: HELLO
from Thread-7: HELLO
from Thread-6: HELLO
from Thread-2: HELLO
from thread Thread-0: Hello, you are client #15.
from thread Thread-0: enter your commands above [kill] to quit
from thread Thread-0:
from thread Thread-0: finished consuming initial message
from thread Thread-3: Hello, you are client #16.
from thread Thread-3: enter your commands above [kill] to quit
from thread Thread-3:
from thread Thread-3: finished consuming initial message
from thread Thread-16: Hello, you are client #19.
from thread Thread-16: enter your commands above [kill] to quit
from thread Thread-16:
from thread Thread-4: Hello, you are client #17.
from thread Thread-16: finished consuming initial message
from thread Thread-4: enter your commands above [kill] to quit
from thread Thread-4:
from thread Thread-4: finished consuming initial message
from thread Thread-1: Hello, you are client #18.
from thread Thread-1: enter your commands above [kill] to quit
from thread Thread-1:
from thread Thread-1: finished consuming initial message
from Thread-0: HELLO
from Thread-3: HELLO
from Thread-16: HELLO
from Thread-1: HELLO
from Thread-4: HELLO
All users have finished
```

And for brevity just an instance of the server increasing the amount of threads is shown below



```
=================CHECKING MAIN THREADS==================
--The number of elements in the main queue is 11--
--ThreadPool has increased the number of main threads the thread count is 15--
--re-assigning main pool--
main thread pool started
======================COMPLETE=========================
```

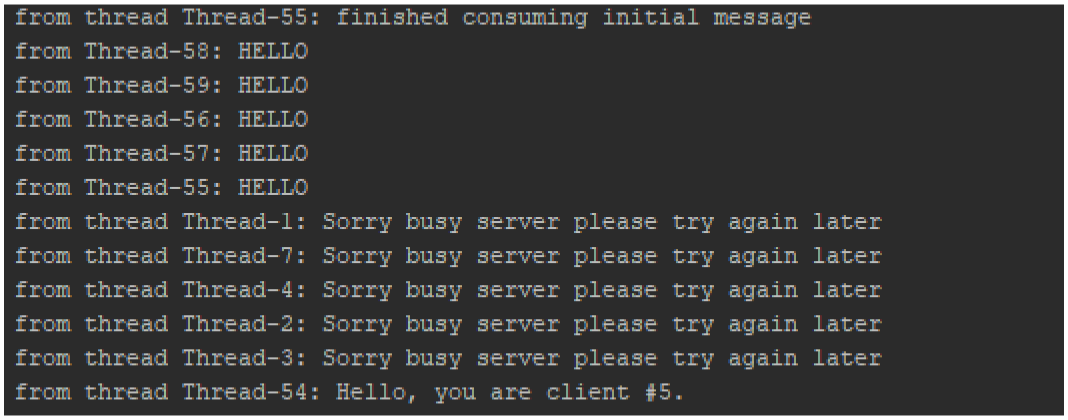
And also an instance of the server decreasing the number of threads as the queue is serviced



```
=================CHECKING MAIN THREADS==================
--The number of elements in the main queue is 10--
--ThreadPool has increased the number of main threads the thread count is 6--
--The number of main threads in use is 3--
--ThreadPool has decreased the number of threads the thread count is 3--
======================COMPLETE=========================
```

The way I designed the methods to perform these changes checks to see the number of elements in the job queue and compares that with how many threads are in actively running if that number isn't high if the number of threads running is less than the total threads it will halve the number of threads. When the timer object checks the threads again when it sees that the thread count is less than the default of five it will reset the total number of threads back to the default.

Below is an example of the server being busy and sending back a busy server message



```
from thread Thread-55: finished consuming initial message
from Thread-58: HELLO
from Thread-59: HELLO
from Thread-56: HELLO
from Thread-57: HELLO
from Thread-55: HELLO
from thread Thread-1: Sorry busy server please try again later
from thread Thread-7: Sorry busy server please try again later
from thread Thread-4: Sorry busy server please try again later
from thread Thread-2: Sorry busy server please try again later
from thread Thread-3: Sorry busy server please try again later
from thread Thread-54: Hello, you are client #5.
```

Issues:

The main issue I had when initially designing my solution was the way I was having the thread pool create threads to service the user. I had originally though that the server would have a thread to service the user's connection and have threads for the different actions on the server i.e the users connection maintained a main thread and then any interaction was carried out on a separate thread. This was an original misunderstanding on my part and was remedied by keeping a pool of threads to service the main connections with those then having auxiliary threads if needed for execution. Both pools are monitored by the timer object that repeatedly checks the queues for size and checks how many threads are currently in use. The timer object is ultimately responsible for the shutdown of the server and the command to kill only shuts the server down if there is only no more threads actively running after the command was received.