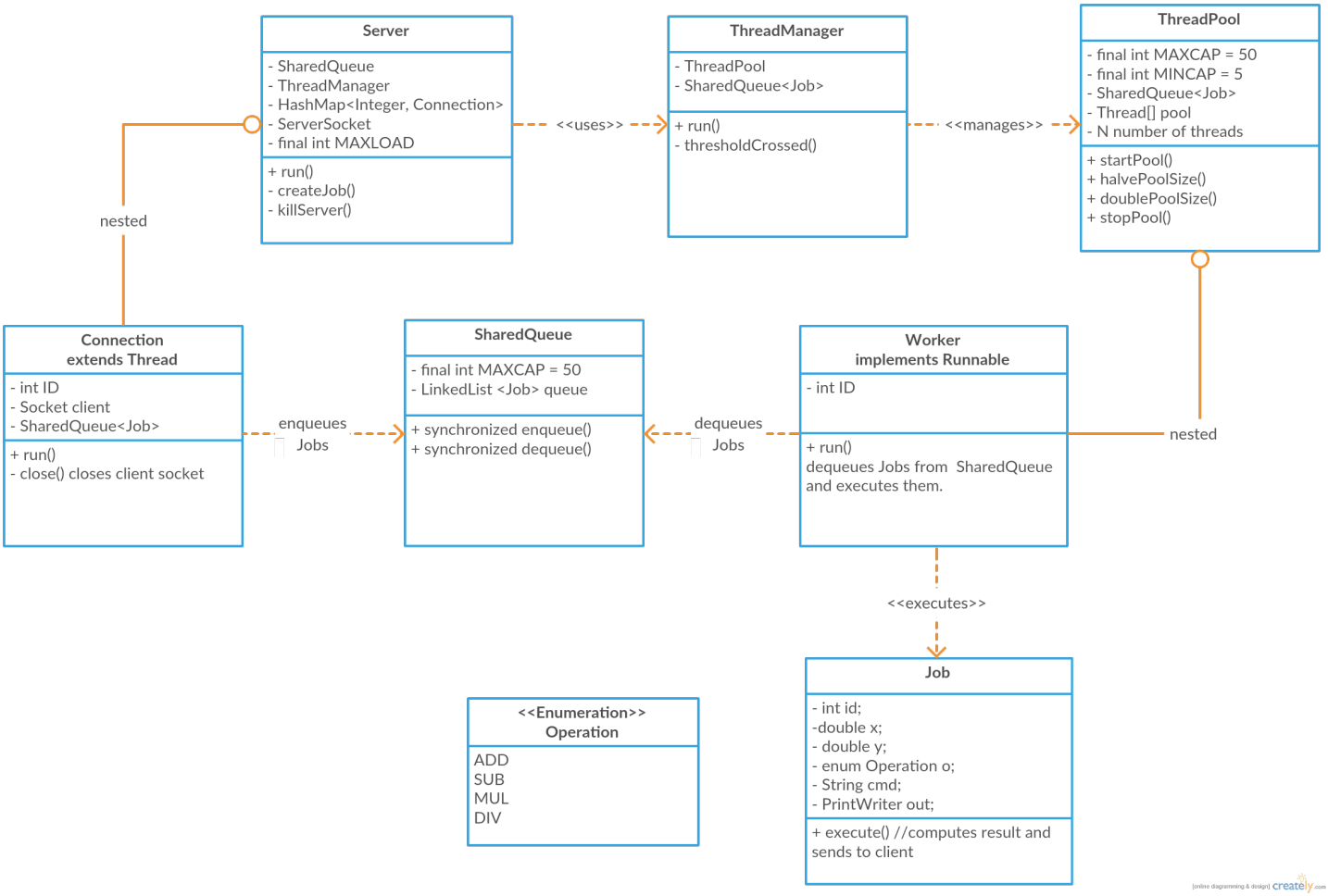
System Design



The server instantiates the following objects:

- HashMap<Integer, Connection>
- ThreadManager
- ThreadPool
- SharedQueue
- ServerSocket

The Server uses an infinite loop to accept up to a MAXLOAD of incoming connections. When a client connects, the Server instantiates a Connection object. The server maintains a HashMap of Connections for fast retrieval and removal.

The Connection object is a thread that uses an infinite loop to accept commands from a client. Commands are parsed and are used to instantiate a Job object, which is then enqueued to the SharedQueue.

The ThreadManager manages the ThreadPool. The ThreadManager runs in a loop until interrupted by the Server. It constantly checks to see if the SharedQueue's size has crossed a threshold. If so, the number of Threads in the ThreadPool is either

doubled or halved depending on whether the SharedQueue is growing or shrinking. Upon interruption, the ThreadManager stops all threads running in the ThreadPool.

The ThreadPool is Thread array of fixed size. The number N Threads running at any given time is controlled by the ThreadManager. Each instantiated Thread contains a Runnable object called Worker. Worker's simply dequeue Jobs from the SharedQueue and execute them.

The SharedQueue is a monitor object with synchronized methods enqueue() and dequeue(). The Connections enqueue Jobs as the clients request them, and the Worker threads dequeue Jobs when available. Otherwise they are forced to wait().

Jobs are simple objects containing an operation and two operands parsed from the client's commands. A call to a Job's execute() method will perform the operation on the operands and send the result back to the client via PrintWriter out stream.

A class called SimpleClient will generate N connections to the server at the IP address provided in the command line arguments. The connections all request the same job: ADD,1,1.

Issues

There were no substantial issues while creating this design. The new code was an extension of the initial code provided. Initially the Server maintained an array of Connections, but it was later deemed better to use a HashMap for added flexibility and quick access when removing and adding new Connections.

Test Results

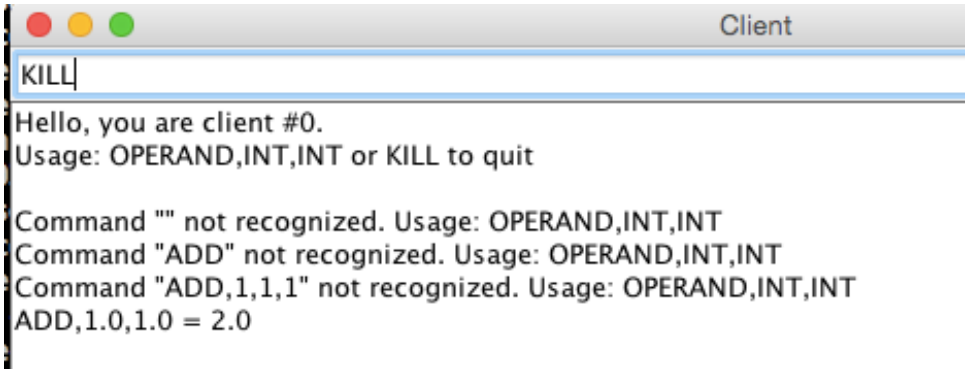
Item 6: Thread pool doubling (lines 2 & 20)

```
New connection with client# 35 at Socket[addr=/10.101.176.150,port=58076,localport=9898] at 2016.02.19.14.01.22
ThreadManager doubled pool size to 10 at 2016.02.19.14.01.22
Worker id=2 processed client:15 job:ADD,1.0,1.0 = 2.0 at 2016.02.19.14.01.22
New connection with client# 36 at Socket[addr=/10.101.176.150,port=58077,localport=9898] at 2016.02.19.14.01.22
Worker id=3 processed client:12 job:ADD,1.0,1.0 = 2.0 at 2016.02.19.14.01.22
Worker id=0 processed client:4 job:ADD,1.0,1.0 = 2.0 at 2016.02.19.14.01.22
Worker id=4 processed client:13 job:ADD,1.0,1.0 = 2.0 at 2016.02.19.14.01.22
New connection with client# 37 at Socket[addr=/10.101.176.150,port=58078,localport=9898] at 2016.02.19.14.01.22
New connection with client# 38 at Socket[addr=/10.101.176.150,port=58079,localport=9898] at 2016.02.19.14.01.22
Worker id=1 processed client:6 job:ADD,1.0,1.0 = 2.0 at 2016.02.19.14.01.22
New connection with client# 39 at Socket[addr=/10.101.176.150,port=58080,localport=9898] at 2016.02.19.14.01.22
Worker id=5 processed client:3 job:ADD,1.0,1.0 = 2.0 at 2016.02.19.14.01.22
Worker id=6 processed client:10 job:ADD,1.0,1.0 = 2.0 at 2016.02.19.14.01.22
Worker id=8 processed client:0 job:ADD,1.0,1.0 = 2.0 at 2016.02.19.14.01.22
Worker id=7 processed client:2 job:ADD,1.0,1.0 = 2.0 at 2016.02.19.14.01.22
Worker id=9 processed client:9 job:ADD,1.0,1.0 = 2.0 at 2016.02.19.14.01.22
New connection with client# 40 at Socket[addr=/10.101.176.150,port=58081,localport=9898] at 2016.02.19.14.01.22
New connection with client# 41 at Socket[addr=/10.101.176.150,port=58082,localport=9898] at 2016.02.19.14.01.22
New connection with client# 42 at Socket[addr=/10.101.176.150,port=58083,localport=9898] at 2016.02.19.14.01.22
ThreadManager doubled pool size to 20 at 2016.02.19.14.01.22
New connection with client# 43 at Socket[addr=/10.101.176.150,port=58084,localport=9898] at 2016.02.19.14.01.22
New connection with client# 44 at Socket[addr=/10.101.176.150,port=58085,localport=9898] at 2016.02.19.14.01.22
```

Item 7: Thread pool halving (line 3)

```
Worker id=10 processed client:44 job:ADD,1.0,1.0 = 2.0 at 2016.02.19.14.01.22
Worker id=9 processed client:45 job:ADD,1.0,1.0 = 2.0 at 2016.02.19.14.01.22
ThreadManager halved pool size to 10 at 2016.02.19.14.01.22
Worker id=3 processed client:47 job:ADD,1.0,1.0 = 2.0 at 2016.02.19.14.01.22
Worker id=2 processed client:46 job:ADD,1.0,1.0 = 2.0 at 2016.02.19.14.01.22
Worker id=0 processed client:48 job:ADD,1.0,1.0 = 2.0 at 2016.02.19.14.01.22
Worker id=4 processed client:49 job:ADD,1.0,1.0 = 2.0 at 2016.02.19.14.01.22
Worker id=8 processed client:50 job:ADD,1.0,1.0 = 2.0 at 2016.02.19.14.01.22
New connection with client# 51 at Socket[addr=/10.101.176.150,port=58100,localport=9898] at 2016.02.19.14.01.23
New connection with client# 52 at Socket[addr=/10.101.176.150,port=58099,localport=9898] at 2016.02.19.14.01.23
Worker id=2 processed client:51 job:ADD,1.0,1.0 = 2.0 at 2016.02.19.14.01.23
```

Item 8: Server handles bad commands and KILL command gracefully




```
pehr037926wp:src Kale$ java Server
The server is running.
New connection with client# 0 at Socket[addr=/10.101.176.150,port=59244,localport=9898] at 2016.02.19.15.14.09
Worker id=4 processed client:0 job:ADD,1.0,1.0=2.0 at 2016.02.19.15.14.27
KILL command received from client# 0 at 2016.02.19.15.14.43
Killing client connections at 2016.02.19.15.14.43
Killing server at 2016.02.19.15.14.43
pehr037926wp:src Kale$
```