

CNT 4007: Programming Assignment #1

Compile

`javac *.java`

Run

[Console 1] `java server [port]`

[Console 2] `java client [serverURL] [port]`

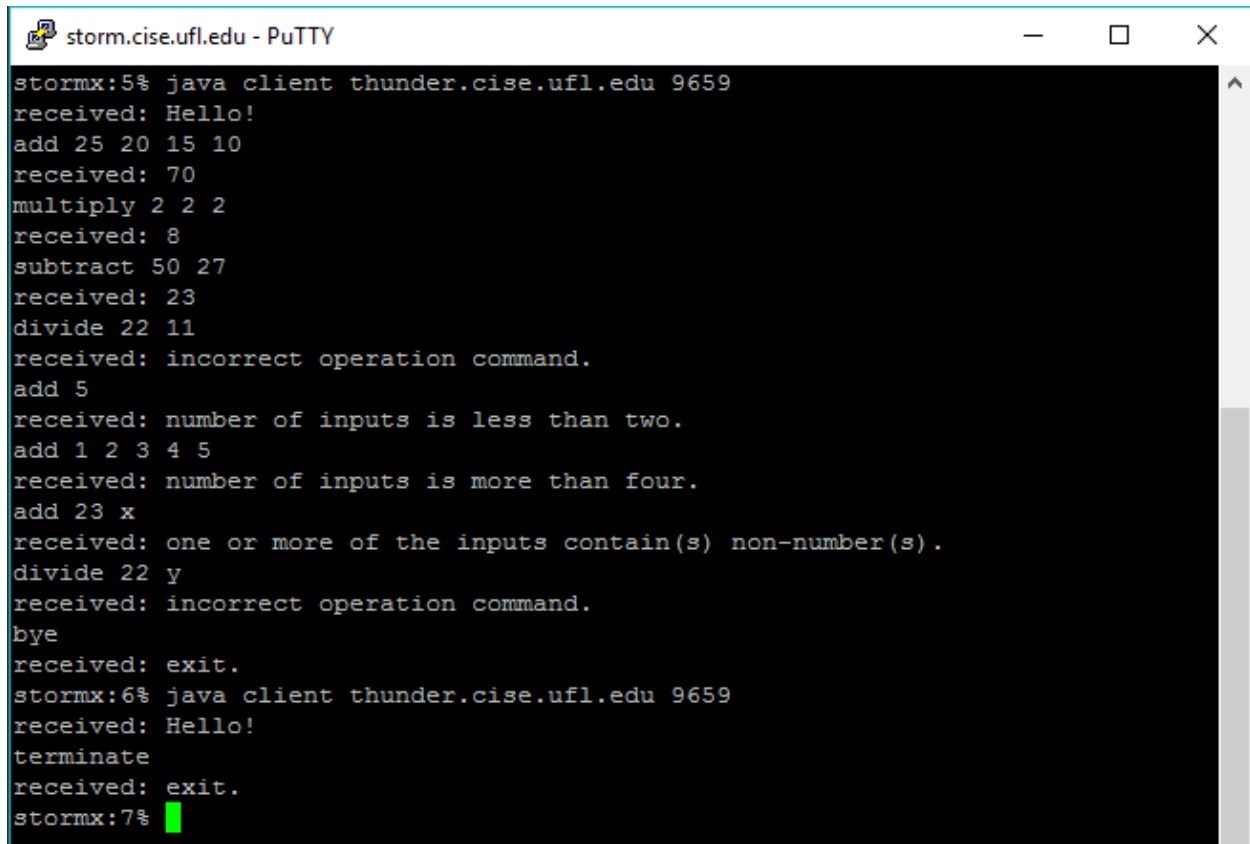
Code Structure

The server program is arranged as a couple of helper functions that are called in main to initialize the listening socket. Two nested try/catches ensure appropriate shutdown of the server and socket upon completion.

The client program initializes a client class in main. The client class holds necessary functions for connecting to the server with the provided server URL and port number. It also features a while loop for reading user input and sending it to the server.

Both programs have helper functions to parse data coming from the socket IO stream.

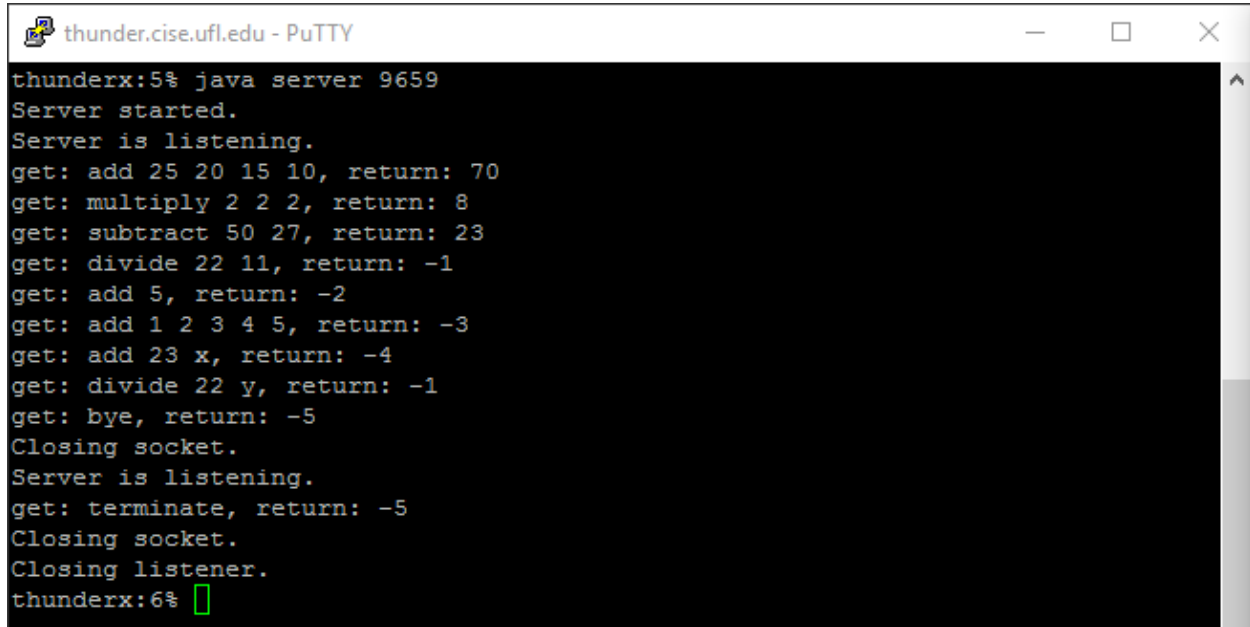
Execution Results



```
stormx:5% java client thunder.cise.ufl.edu 9659
received: Hello!
add 25 20 15 10
received: 70
multiply 2 2 2
received: 8
subtract 50 27
received: 23
divide 22 11
received: incorrect operation command.
add 5
received: number of inputs is less than two.
add 1 2 3 4 5
received: number of inputs is more than four.
add 23 x
received: one or more of the inputs contain(s) non-number(s).
divide 22 y
received: incorrect operation command.
bye
received: exit.
stormx:6% java client thunder.cise.ufl.edu 9659
received: Hello!
terminate
received: exit.
stormx:7% █
```

Client log on storm.cise.ufl.edu

We see the various requirements of the assignment exemplified above. The server accepts input of the form <operation> <inputs[2-4]> and provides errors for incorrect operations or wrong number of inputs as well as non-numeric inputs. We also see the server remains active when receiving “bye” from the client, but closes along with the client when the client issues a “terminate.”



```
thunderx:5% java server 9659
Server started.
Server is listening.
get: add 25 20 15 10, return: 70
get: multiply 2 2 2, return: 8
get: subtract 50 27, return: 23
get: divide 22 11, return: -1
get: add 5, return: -2
get: add 1 2 3 4 5, return: -3
get: add 23 x, return: -4
get: divide 22 y, return: -1
get: bye, return: -5
Closing socket.
Server is listening.
get: terminate, return: -5
Closing socket.
Closing listener.
thunderx:6%
```

Server log on thunder.cise.ufl.edu

Looking at the returns it should be clear that computation is handled properly. Also shown is the required error codes as well as a case where a lower number error code takes precedence over the higher number code (-1 is returned for the command that also had a -4 error). Finally, you can see where the client sent “bye” and the socket was closed, but the server did not and subsequently another client connected and the server began listening again. Upon receiving “terminate” both client and server closed.

Bugs

None

Comments

None