Assignment 4: Chat Server

Out: August 26, 1999
Due: September, 1999
Extension: September, 1999

Implement a chat server for use on the Internet.

1 Invocation

The server should be started by the command line:

```
java ChatServer port
```

or the equivalent of your Java platform. The parameter *port* is the port number the server will attach itself to. If it is not given, use 7777 as the port number.

Once started, the server will not stop.

2 The Protocol

We will use the following simple protocol.

Connection phase: When a client connects to the server, it will send a username. A username must not contain colons. If the username is already in use by another client, the server will send the character `N` to the client indicating refusal, and the client may try another username or disconnect. If eventually the client sends a username not already in use, the server will send the character `Y` indicating acceptance. From this point onwards we enter the chat phase.

Chat phase: When a client sends a line[1] to the server, the server will look up the username of the client, prepend it and a colon to the line, and broadcast the result to all clients (including the originator). Example: `monochrom` sends the line

```
Hello everyone: how are you?
```

[1] A line must end with the newline character.

to the server. The server then broadcasts the following line to all clients:

```
monochrom:Hello everyone: how are you?
```

The client can disconnect anytime.

Optional liveness test: The server may choose to test if a client is alive. To do so, the server will send the string `PING` to the client, and expect it to respond with the string `PONG` within a reasonable period. If the client does not respond, the server may disconnect the client.

3 Robustness

The server should have these basic robustness:

- It must not crash.
- It must not go into a deadlock.
- It must be responsive to clients, i.e., no starving.

Bonus will be given to a server that also withstands denial-of-service attacks, such as:

- a client that does not finish the connection phase within a reasonable amount of time or number of trials
- a client that sends too many stuff in too short a time (flooding)
- a client that never reads
- a site that creates too many connections