

```
package myChat;

/**
 *
 * @author Patrick Neubauer
 * @version 1.0 alpha
 *
 * Course: Distributed Systems
 * Project P1: Internet Chat System
 *
 * Team members:
 * Tenzin Dakpa, Patrick Neubauer
 *
 */
import java.net.*;
import java.io.*;

public class TCPServer implements Runnable {

    // defining private properties
    private int clientCounter = 0;

    private ServerSocket server = null;

    private Thread thread = null;
    // clients[] is an array that saves all the clients connected to the server
    // clients are TCPServerThread's
    private TCPServerThread clients[] = new TCPServerThread[50];

    /*
     * constructor for the server setting up on starting it.
     */
    public TCPServer(int port) {
        try {
            System.out.println("Server: doing the set up ...");
            System.out.println("Server: setting port at " + port);
            server = new ServerSocket(port);
            System.out.println("Server: started server on " + server);
            start(); // create new thread for client
        } catch (IOException ioe) {
            System.out.println("Server: cannot start server at port " + port + ".\n Error: " + ioe.getMessage());
        }
    }

    /* creating a new thread and starting it
     */
    public void start() {
        if (thread == null) {
            thread = new Thread(this);
            thread.start();
        }
    }

    /* stop a thread
     */
    public void stop() {
        if (thread != null) {
            thread.stop();
            thread = null;
        }
    }

    /* start waiting for clients, accepting and adding new threads

```

```

public void run() {
    while (thread != null) {
        try {
            System.out.println("Server: waiting for clients ...\n");
            addThread(server.accept());
        } catch (IOException ioe) {
            System.out.println("Server: input/output error: " + ioe + "\n");
            stop();
            System.out.println("Server: stopped server.\n");
        }
    }
}

/* findClient
*/
private int findClient(int ID) {
    for (int i=0; i<clientCounter; i++) {
        if (clients[i].getID() == ID)
            return i; // returns the position of the client with a certain ID
    }
    return -1; // if client ID not found
}

/* handle messages, broadcast it to all connected clients
synchronized
*/
public synchronized void handle(int ID, String input) {
    if (input.contains("/exit")) {
        // find the client ID in the array of TCPServerThread clients
        // and send him the leaving command
        clients[findClient(ID)].send("/exit");
        remove(ID); // finally remove the client out of the array
    } else {
        // send all the connected clients the input from the client,
        // broadcast messages to all connected clients ...
        for (int i=0; i<clientCounter; i++) {
            clients[i].send(input);
        }
    }
}

/* terminate client and remove it from server
*/
public synchronized void remove(int ID) {
    int pos = findClient(ID); // get position
    if (pos >= 0) {
        // get the client out of the array that has to be terminated
        TCPServerThread toTerminate = clients[pos];
        System.out.println("Server: TERMINATING CLIENT thread with ID " + ID + " at position " + pos);
        if (pos < clientCounter-1) {
            for (int i=pos+1; i<clientCounter; i++) {
                clients[i-1] = clients[i];
            }
            clientCounter--; // decrease clientCounter after removed client
        }
        try {
            toTerminate.close(); // closes socket and data streams
        } catch (IOException ioe){
            System.out.println("Server: error while closing thread: " + ioe);
        }
        toTerminate.stop(); // stop client thread
    }
}

```

```

    }
}

/* add new tcp server threads to the client array on server
*/
private void addThread(Socket socket) {
    if (clientCounter < clients.length) {
        System.out.println("Server: new CLIENT ACCEPTED on " + socket);
        // add new client to TCPServerThread array of clients
        clients[clientCounter] = new TCPServerThread(this, socket);
        try {
            clients[clientCounter].open(); // open connection to client
            clients[clientCounter].start(); // starting connection to client
            clientCounter++;
        } catch (IOException ioe) {
            System.out.println("Server: error while opening thread: " + ioe);
        }
    } else {
        System.out.println("Server: client refused, because maximum length of " + clients.length + " reached.");
    }
}

public static void main (String args[]) {
    TCPServer server = null;
    int port;
    port = 7896;
    server = new TCPServer(port);
}
}

```