CS 4390 Computer Networks

Chat Program Project Write-up

Alex Szeto – acs140530

The objective of this project was to design and to program, using socket programming, a simple chat session application that allows for two people to chat with each other. The program that I developed is run on the command console, and allows two people, and theoretically even more people to communicate in a group chat together, with each person’s message broadcasted to the other people paired with a unique identifier. The program will also notify the other users if a user disconnects, and the users will also be notified if the server was suddenly disconnected.

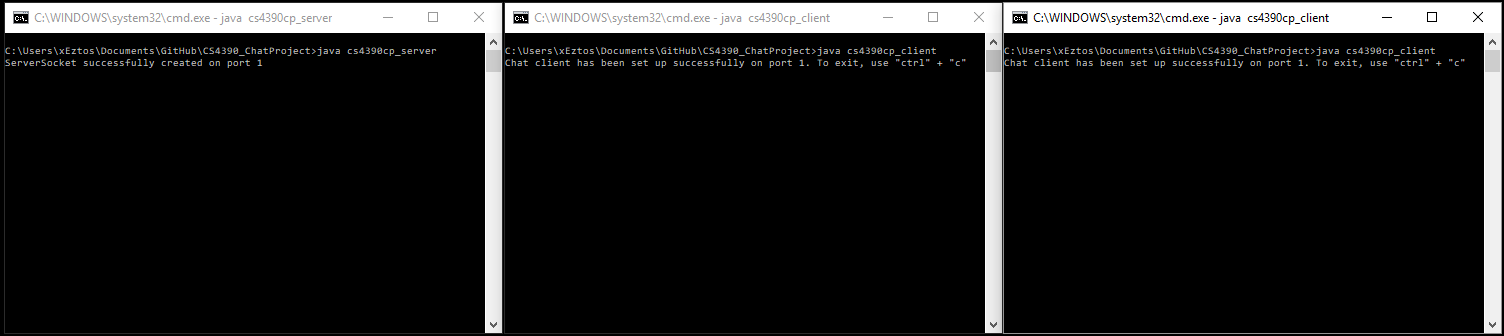
An initial hiccup that I ran into was the fact that I did not understand what the term “socket programming” meant, since I have never heard the term before. A quick google of the term revealed that the term merely referred to the idea that programs were talking to using open listening port sockets. I then did much research into the java.net networking API.

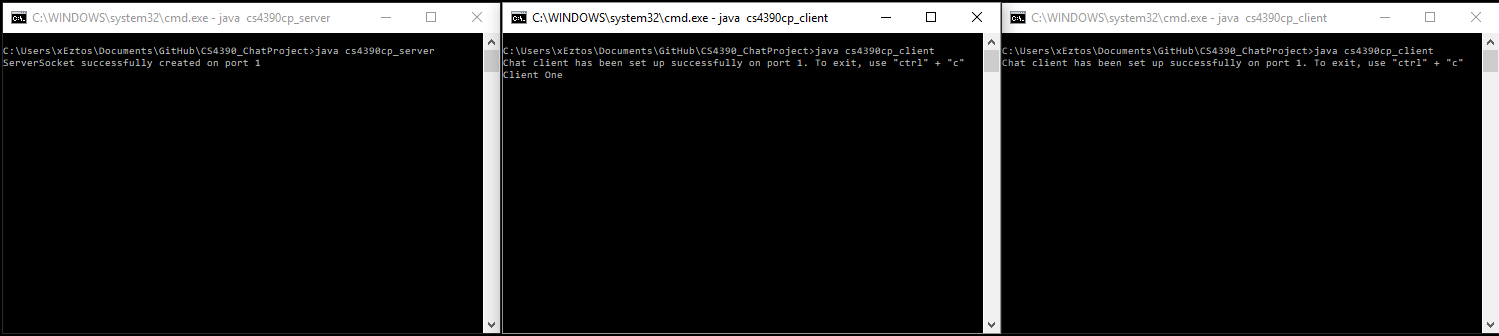
Initially, the server did not have

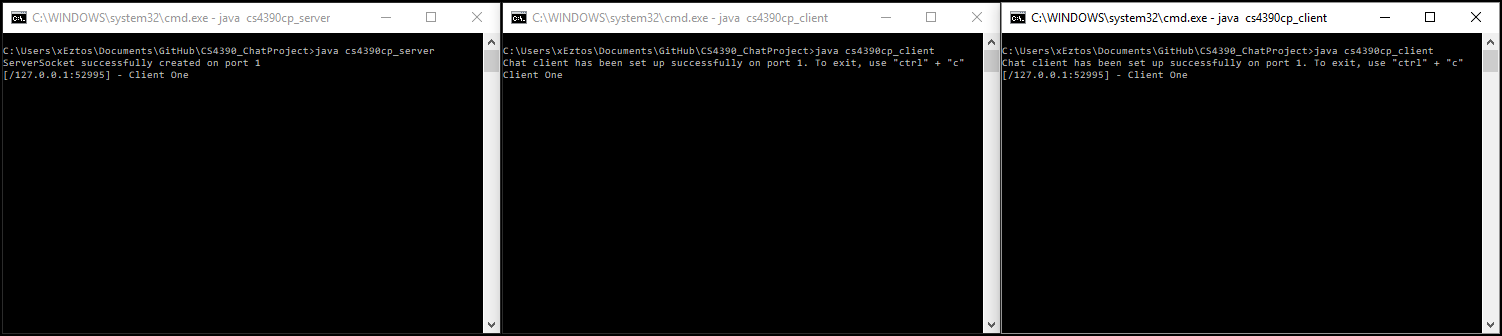
After I got a general framework hammered out, I very quickly realized that the client-side program was not able to both listen to the server messages and listen to the user console messages at the same time. Luckily, the projects in CS 4348 Operating Systems helped me realize that if the monitoring of both server-side messages and client-side messages were handled on separate threads within the same program, they would be able to run concurrently. That lead to the “extend” of the thread in the server-side monitoring client. The Java Scanner package already runs in its own thread, meaning that I did not have to implement the scanner as its own thread.

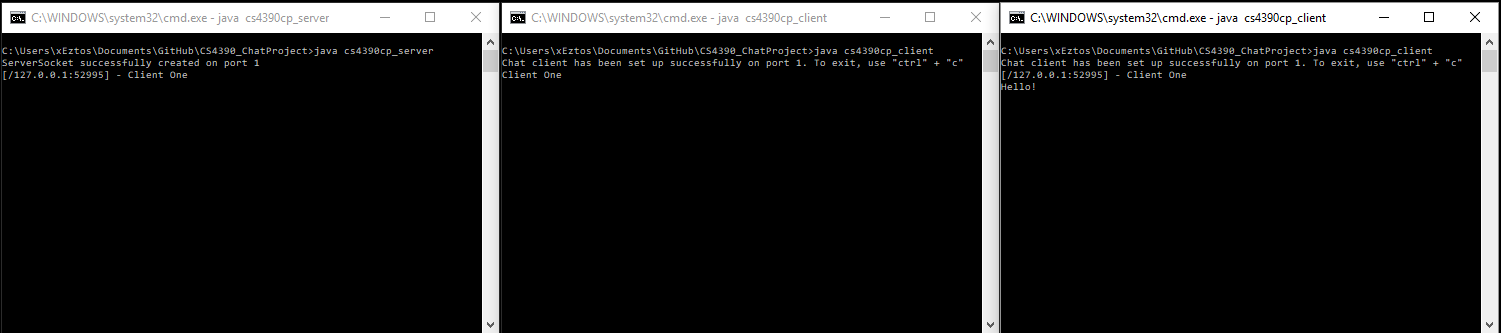
The following series of screen captures represent a sample program execution:





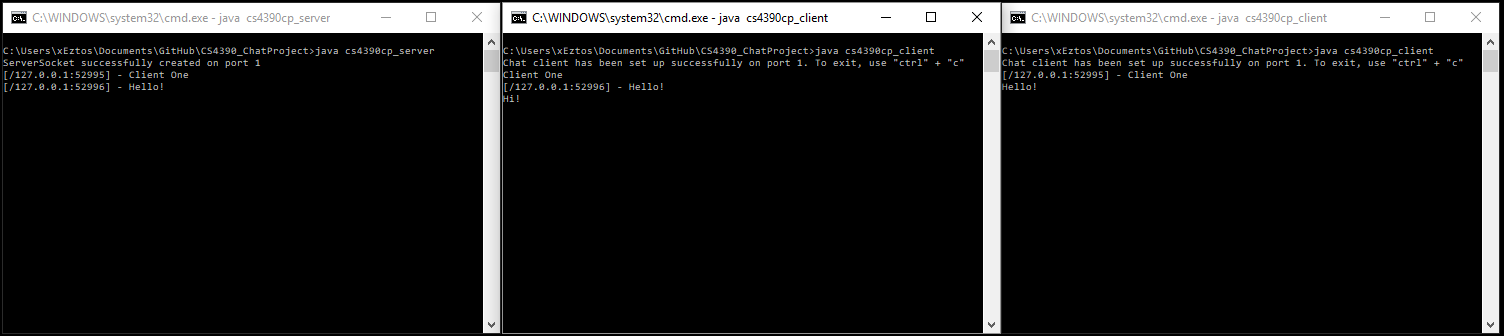


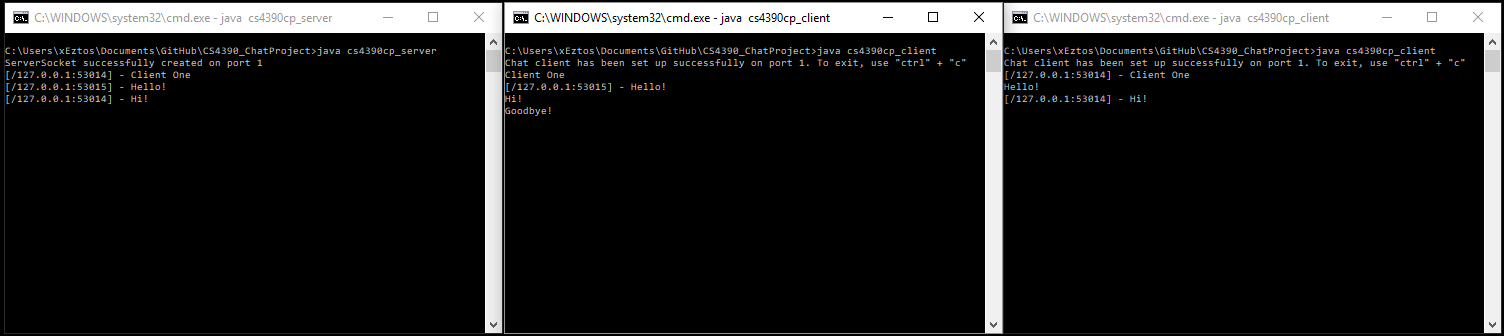


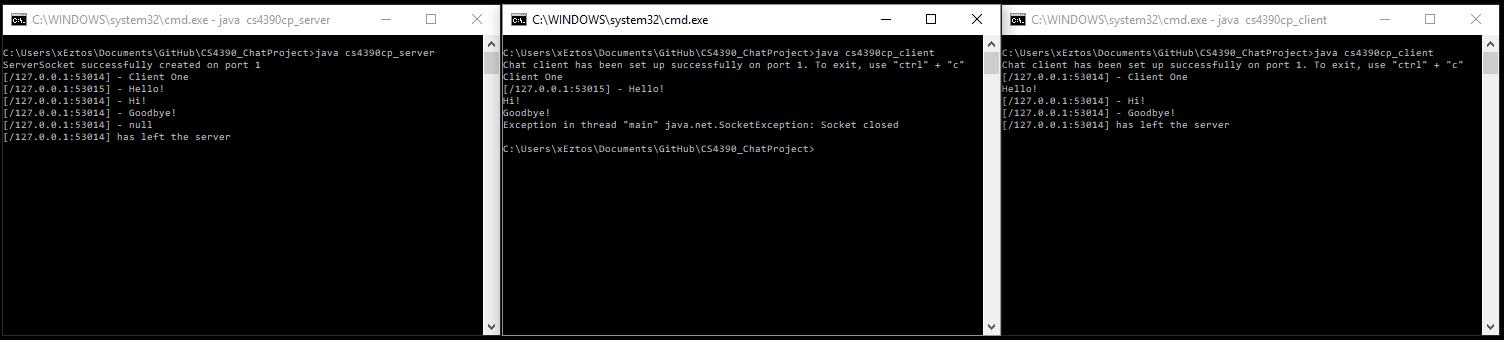


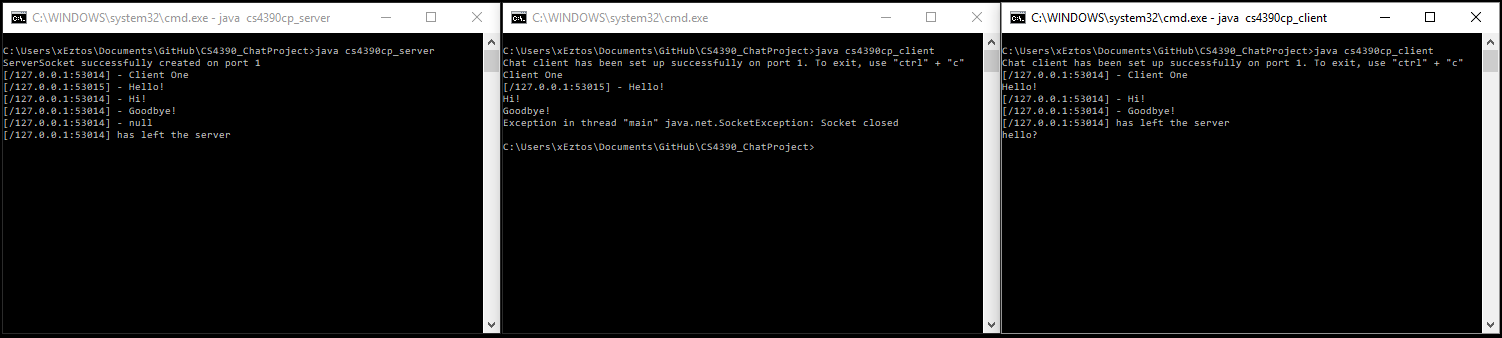


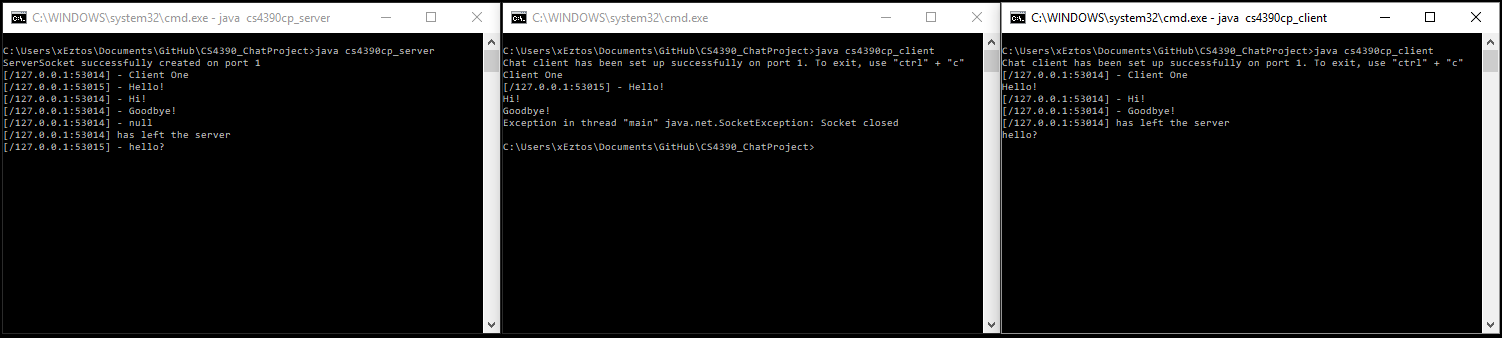


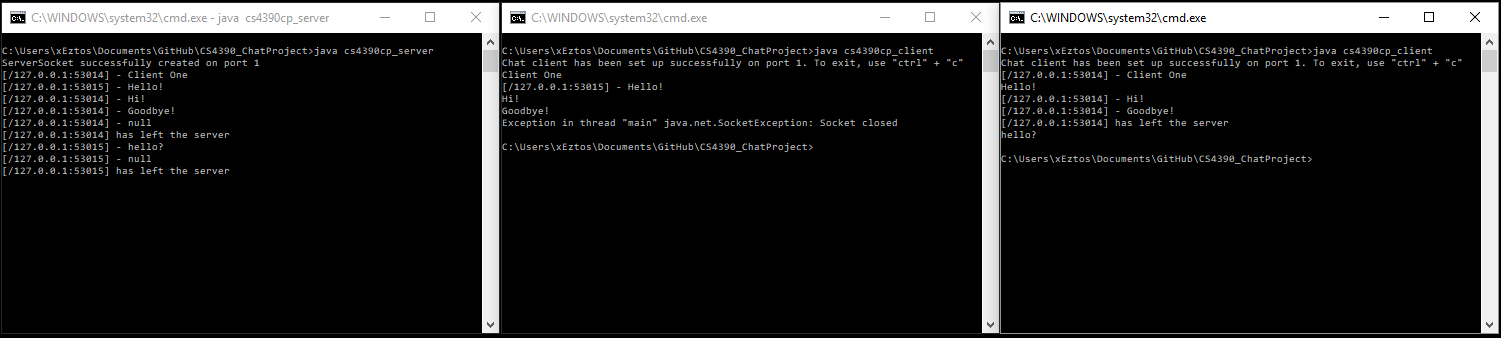


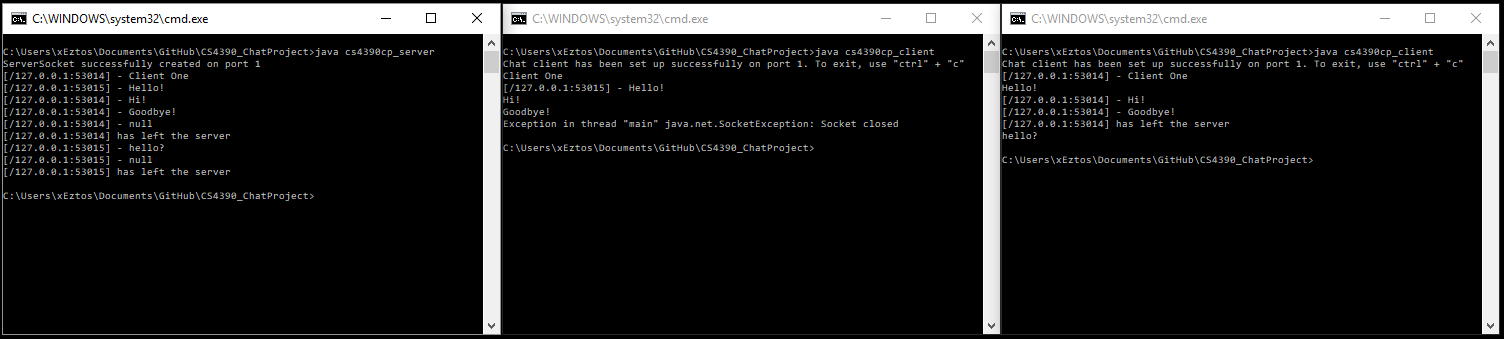






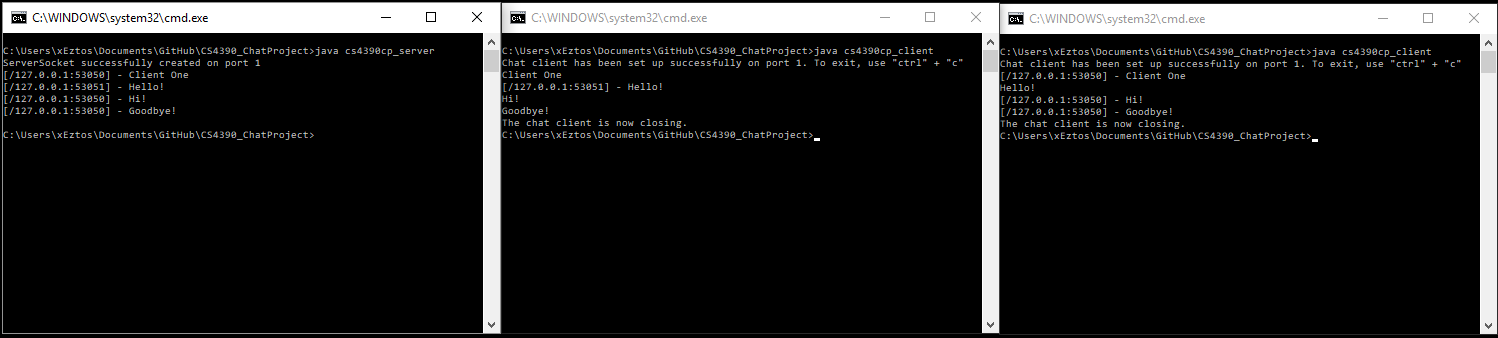






These Two images represent the server closing while there are clients connected:





Server Code:

import java.net.\*;

import java.util.\*;

import java.io.\*;

public class cs4390cp\_server{

static int port = 1;

static ArrayList<PrintWriter> pw = new ArrayList<PrintWriter>();

public static void main(String[] args) throws IOException{

ServerSocket socket = null;

try{

socket = new ServerSocket(port);

System.out.printf("ServerSocket successfully created on port %d%n", port);

} catch (IOException e){

System.out.println("ERROR FLAG 5");

System.exit(-1);

}

try{

while(true){

new ClientComm(socket.accept()).start();

}

} finally{

try{

socket.close();

} catch (Exception e){

System.out.println("ERROR: FLAG 0");

System.exit(-1);

}}}

private static class ClientComm extends Thread{

Socket socket;

BufferedReader input;

PrintWriter output;

ClientComm(Socket socket){

this.socket = socket;

}

public void run(){

try{

input = new BufferedReader(new InputStreamReader(socket.getInputStream()));

output = new PrintWriter(socket.getOutputStream(), true);

pw.add(output); // adds a client PrintWriter into an arrayList of printwriters

while(true){

String temp = input.readLine();

String ret = String.format("[%s:%d] - %s",socket.getLocalAddress().toString(),socket.getPort(),temp);

System.out.println(ret);

if(temp == null){

String message = String.format("[%s:%d] has left the server", socket.getLocalAddress().toString(), socket.getPort());

System.out.println(message);

broadcast(message);

return;

} else if(pw.size() == 1){

output.printf("SERVER - There is nobody else on the server.");

}else{

for(int i = 0; i < pw.size(); i++){

PrintWriter tempW = pw.get(i);

if(tempW != output){

tempW.println(ret);

}}}}

} catch(IOException e){ // client forcefully (ctrl + c) disconnects

System.out.println("ERROR: FLAG 1");

return;

} finally{

pw.remove(output);

try{

socket.close();

} catch( IOException e){

System.out.println("ERROR: FLAG 4");

}}}

public void broadcast(String s){

for(int i = 0; i < pw.size(); i++){

PrintWriter tempW = pw.get(i);

tempW.println(s);

}}}}

Client Code:

import java.io.\*;

import java.util.\*;

import java.net.\*;

public class cs4390cp\_client{

static int port = 1;

static Socket socket;

static PrintWriter toServer;

static BufferedReader fromServer;

public static void main(String[] args) throws IOException{

try{

try{

socket = new Socket("localhost", port);

} catch (UnknownHostException e){

System.out.println("ERROR: cs4390cp\_client FLAG 0");

System.exit(-1);

} catch (IOException e){

System.out.println("ERROR: cs4390cp\_client FLAG 1");

System.exit(-1);

}

toServer = new PrintWriter(socket.getOutputStream(), true);

new serverRecieve().start();

Scanner sc = new Scanner(System.in);

System.out.println("Chat client has been set up successfully on port " + port + ". To exit, use \"ctrl\" + \"c\"");

while(true){

String temp = sc.nextLine();

toServer.println(temp);

}} finally{

socket.close();

}}

private static class serverRecieve extends Thread{

public void run(){

try{

fromServer = new BufferedReader(new InputStreamReader(socket.getInputStream()));

while(true){

String temp = fromServer.readLine();

if(temp == null){

try{

wait();

} catch (InterruptedException e){

System.out.println("ERROR: serverSend FLAG 1");

}} else{

System.out.println(temp);

}}} catch (IOException e){

System.out.printf("The server has been disconnected. The client is now closing.");

System.exit(-1);

}}}}