**Socket Programming Project Report - 4/15/19**

**By Kathleen Graham**

A socket is one endpoint of a two-way communication link between two programs running on the network. A socket is bound to a port number on the network. A socket is bound to a port number so that the TCP layer can identify the application that data is destined to be sent to. An endpoint is a combination of an IP address and a port number. Normally, a server runs on a specific computer that is bound to a specific port number. The server just waits, listening to the socket for a client to make a connection request. Then, on the client side, the client knows the hostname of the machine on which the server is running and the port number on which the server is listening. To make a connection request, the client tries to meet with the server on the server’s machine and port. The client also needs to identify itself to the server so it binds to a local port number that it will use during this connection. Once the server accepts the connection, the server gets a new socket bound to the same local port and has its remote endpoint set to the address and port of the client. It needs a new socket so that it can continue to listen to the original socket for connection requests while tending to the needs of the connected client. On the client side, if the connection is accepted, a socket is successfully created, and the client can use the socket to communicate with the server.

**Methods for PalindromeChecker:**

* Main (string [] args) – Handles the message from the server
* Client\_NewMessageEvent (object sender, byte[] data) - gets the string data

**Methods for PalindromeCheckerServer:**

* Main (string [] args) – handles parameters from the command line and creates the server
* Server\_NewClientEvent(object sender, object newClient) – Handles new connection requests
* Server\_NewMessageEvent (object sender, object client, byte[] data, ) – Checks if the message is a palindrome and tells the user whether or not it is a palindrome

**Methods for MyTcpClient:**

* MyClient (TcpClient client, bool isConnected) - Initialize a new TcpClient from existing built in TcpClient, and reads the data from the user and sends the received data to the handler then end
* MyClient – creates new custom client
* Connect(IPAddress ip, int port) - Connects to the server
* Dispose() - when the client is ready to be closed, it will close.
* Send(byte[] data) - sends the data
* SendString(string data) - sends the string

**Text for the usermanuals:**

**Client**

1. Use "./PalindromeChecker.exe -ip=127.0.0.1 -port=1200" to run the program. (Run the server first!)

2. You can run multiple instances to test the server that can handle more than one clients.

3. You can send an empty string to tell server the client is about to be closed.

4. When all instances are closed, the active users of server will be 0. Then, the server will be closed automatically.

Use a terminal to run the program, don't just double click!

If you are using Ubuntu, don't add '.exe' at the end of the program.

**Server**

1. Use "./PalindromeCheckerServer.exe -port=1200" to run the program. (Run this first before you run clients!)

2. You can run multiple clients to test the server that can handle more than one clients.

3. You can send an empty string from clients to tell the server that the clients are about to be closed.

4. When all clients are closed, the active users of server will be 0. Then, the server will be closed automatically.

Use a terminal to run the program, don't just double click!

If you are using Ubuntu, don't add '.exe' at the end of the program.

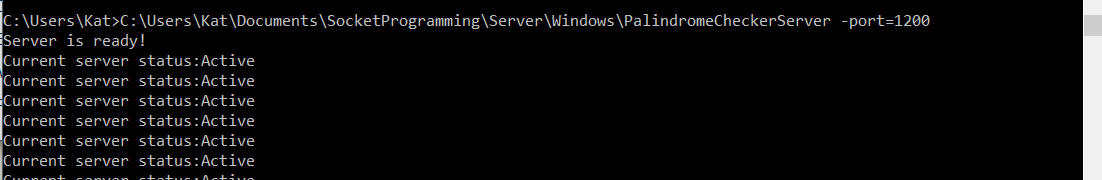
**Tests**

**Local Host Test**



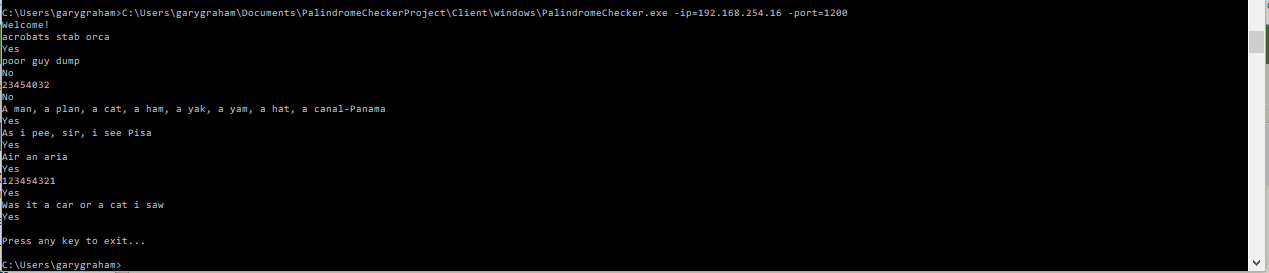
**LAN Test On two different computers**

**My laptop:**





**Desktop:**



**GitHub Link:** [**https://github.com/yliao1/PalindromeChecker**](https://github.com/yliao1/PalindromeChecker)

I was the “CIS person” in this group, so I oversaw the planning and the testing. My programming partner did all the programming, he finished it quickly, and the code was flawless. Most people would say that this would be a dream come true, but I wish that I could’ve done a little more to contribute to the project. I understand the premise of the project and I understand what he did, but I feel like it would have been better to work together in person and understand his thought process.

Testing the program was very easy, all I had to do was go into command prompt and run the server first with the correct file path and then run another command prompt with the client file pathway. Then the client would prompt the user to enter a string. Upon entering the string, the program would spit out a “Yes” or a “No” to whether the string was a palindrome or not. As I said previous, the results were correct, and the program ended whenever the string entered was NULL.

The two String tests that I used were “Acrobats stab orca” and “poor guy dump.” Both tests worked adequately, as did the other strings given in the assignment. I believe that this assignment was nice, but I felt that the workload was bound to be more one-sided. I feel that maybe the assignment could have required both to presently work together in a scheduled class lab to set aside the issue of schedule conflicts. All in all, I did enjoy the assignment and learned a lot. Before, I didn’t know much about socket programming and how it worked, and I have learned a lot from this experience.