Carl Derline
Karen Bullinger
CSCE 320
3-9-15
Server Report

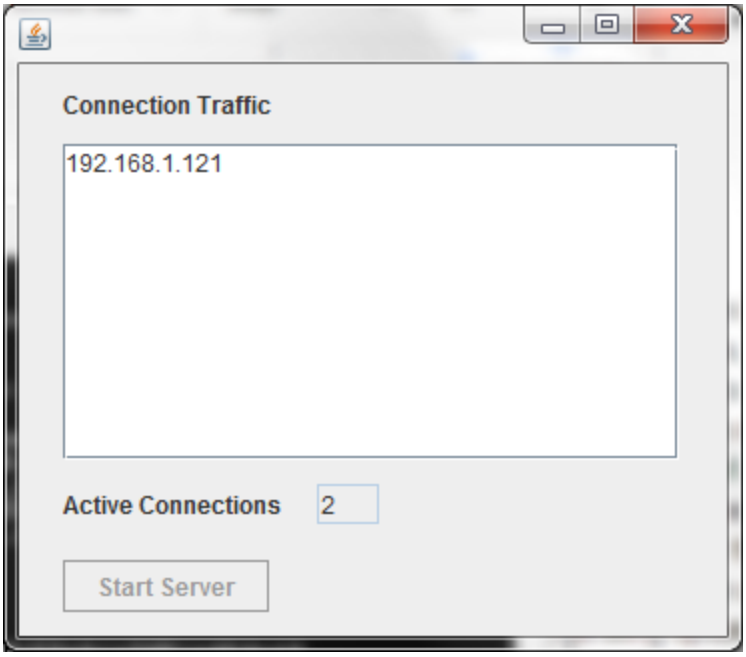
This program is the server side of the echo client-server application. This server handles all client connections, receives messages from the clients, and sends any message sent from a client to the server to each client currently connected to the server.

The program is structured using Model-View-Controller (MVC). A model (serverModel) holds an ArrayList of all current client connections and manages that list by adding or removing clients as they connect or disconnect from the server. The controller (serverController) does most of the work for the application. It is responsible for starting the server (after initiated by a button click from the user), accepting new clients, broadcasting received messages to clients, communicating updates to the model, and communicating updates to the view. Lastly, the view class (serverView) is responsible for receiving updates from the controller and in turn updating the GUI. The view tracks the current number of active connections and the IP addresses of each client that sends a message.

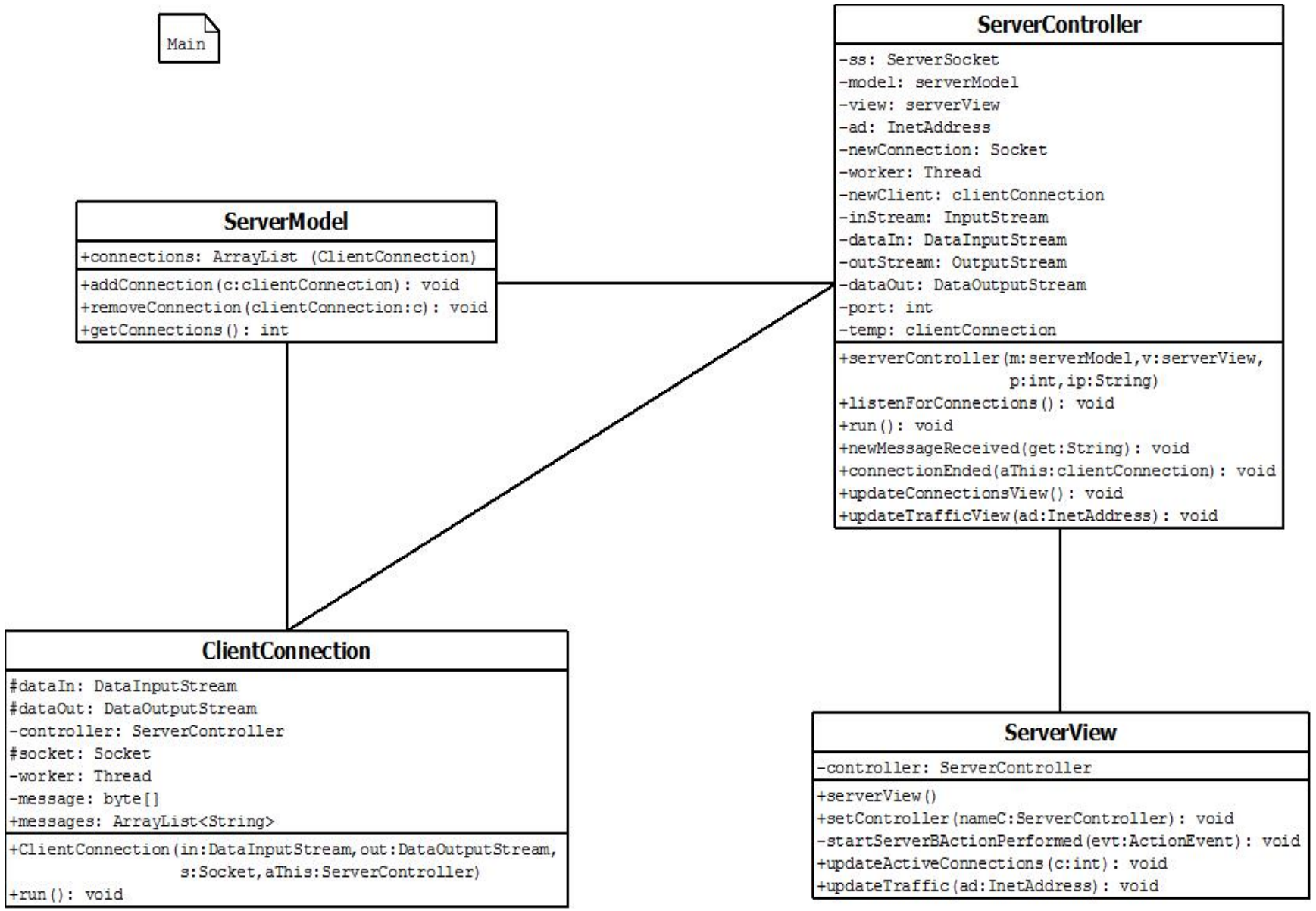
As a way to make managing client connections to the server easier, there is a helper class called ClientConnection. Each time the controller accepts a new connection, this class creates an object that contains all of the necessary data for the client connection (socket, in stream, out stream, controller). Additionally, each time a ClientConnection object is created, it initiates a new thread associated with that object, allowing for the program to constantly listen for new messages from that specific client. This allows for the server to receive several messages at the same time from different clients without crashing the server or creating delays.

When a client disconnects, the program recognizes this and removes that connection from the list of current connections. The view is updated to reflect this each time.

To further improve on this program, the addition of a Stop Server function in the GUI would be convenient. Currently, the server is stopped only when the program is closed by the user.



Server has been started (button is now disabled to prevent user from trying to start the server twice). There are currently two active connections and only one client user has sent a message.


Main
ServerController
-ss: ServerSocket
-model: serverModel
-view: serverView
-ad: InetAddress
-newConnection: Socket
-worker: Thread
-newClient: clientConnection
-inStream: InputStream
-dataIn: DataInputStream
-outStream: OutputStream
-dataOut: DataOutputStream
-port: int
-temp: clientConnection
+serverController(m:serverModel,v:serverView, p:int,ip:String)
+listenForConnections(): void
+run(): void
+newMessageReceived(get:String): void
+connectionEnded(aThis:clientConnection): void
+updateConnectionsView(): void
+updateTrafficView(ad:InetAddress): void
ServerModel
+connections: ArrayList (ClientConnection)
+addConnection(c:clientConnection): void
+removeConnection(clientConnection:c): void
+getConnections(): int
ClientConnection
#dataIn: DataInputStream
#dataOut: DataOutputStream
-controller: ServerController
#socket: Socket
-worker: Thread
-message: byte[]
+messages: ArrayList<String>
+ClientConnection(in:DataInputStream,out:DataOutputStream, s:Socket,aThis:ServerController)
+run(): void
ServerView
-controller: ServerController
+serverView()
+setController(nameC:ServerController): void
-startServerBActionPerformed(evt:ActionEvent): void
+updateActiveConnections(c:int): void
+updateTraffic(ad:InetAddress): void