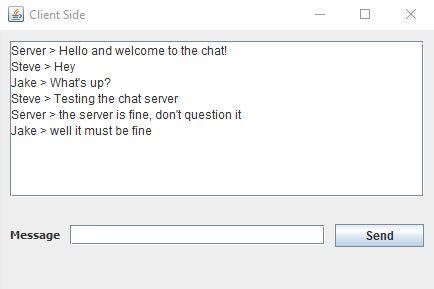
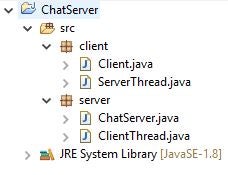
**Creating a Chat Server Using Java**

[](https://cdn.instructables.com/ORIG/FSS/K8A1/ILJWGR24/FSSK8A1ILJWGR24.jpg?auto=webp&frame=1&fit=bounds&md=4dbc8dbe689278ad7708136c9ee2af9e)

[](https://cdn.instructables.com/ORIG/FAT/IZK6/IM55XZJ3/FATIZK6IM55XZJ3.jpg?auto=webp&frame=1&fit=bounds&md=8b4cbfb5922d97b819862336de7255b1)

**Introduction**

Many programming languages have multiple ways to form connections users and servers or between peers. Java can handle network connections through the use of Java Sockets. This tutorial is about how to use sockets in java by developing a chat server between one server and many users.

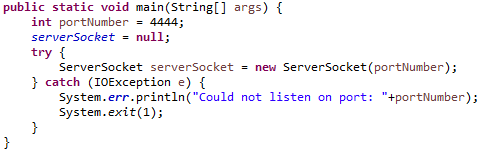
This tutorial is divided between client side and server-side development.

**Requirements:**

* Java JDK (I used version 1.8.0\_74 for this)
* Java IDE (I used Eclipse Mars for this)
* A package structure with the above classes

**Timer:**About 15-30 minutes

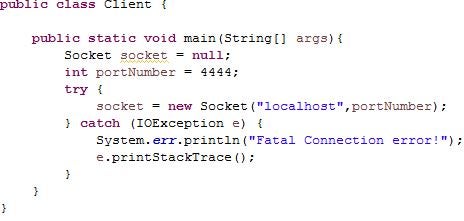
**Step 1: Setup a ServerSocket in the Server Class**



Create a static Server socket at the beginning of main in the ChatServer class and instantiate it with a port number.

TIP: ServerSockets may be instantiated with a port number other than 4444, but numbers lower than 1000 are more likely to already be in use by the system.

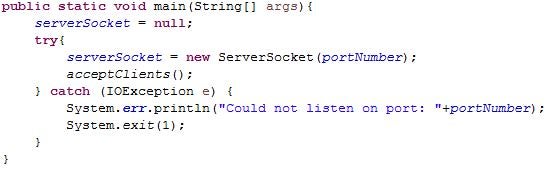
**Step 2: Create a Socket in the Login Class**

[](https://cdn.instructables.com/ORIG/FHH/A9GO/IM55XBVN/FHHA9GOIM55XBVN.jpg?auto=webp&frame=1&fit=bounds&md=a01cb62442954c6f2fd8813d4f9444e1)

Create a socket in the main method of Login, specifying the host address and port number. Since this is a login window, the socket can be an instance variable as it will be passed to the actual chat window later.

TIP: "localhost" is used in the socket parameters for debugging purposes to specify that the server is running on the same computer as the client. Later this can be changed to take a host address to connect to a remote server.

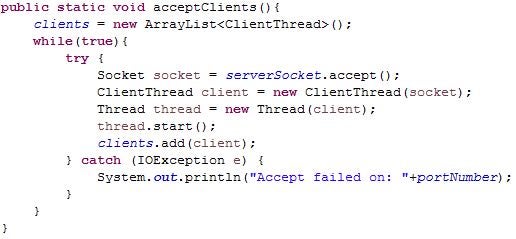
**Step 3: Create a Loop to Continuously Accept Clients**

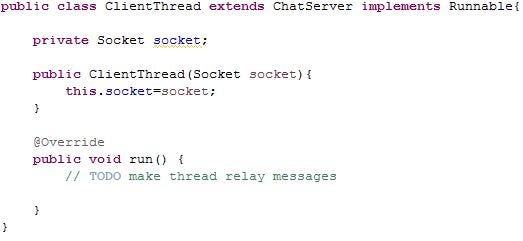




After instantiating the serverSocket in chatServer, start a while loop that continuously accepts all clients.

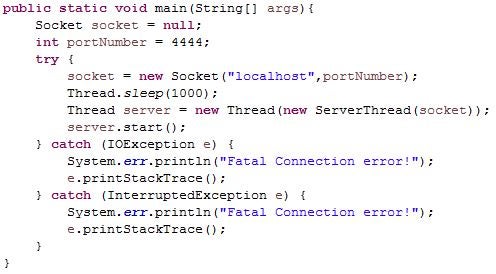
**Step 4: Create the Client Threads**

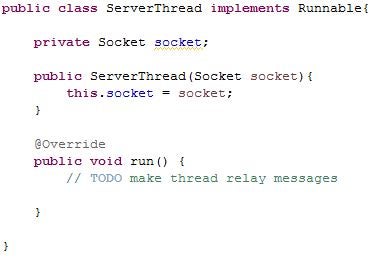




1. With the socket received from serversocket. accept, create a new ClientThread .
2. Create a new Thread using the ClientThread, and then call start on the thread.
3. In ClientThread, make ClientThread extend ChatServer, and implement Runnable
4. Create the ClientThread constructor and add the method run.

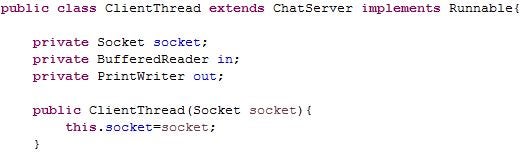
**Step 5: Create the Server Thread**

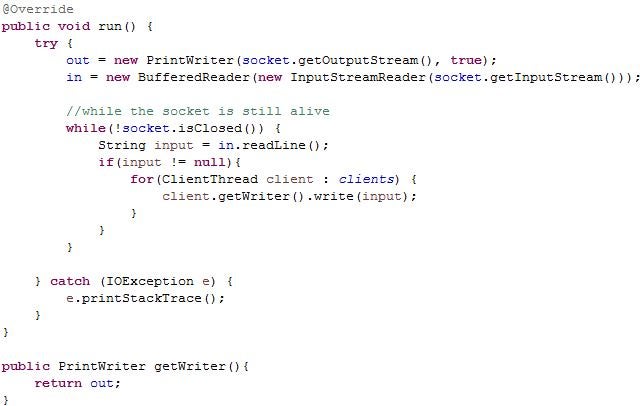




1. Create a new ServerThread using socket.
2. Create a new Thread using the ServerThread, and then call start on the thread.
3. In ServerThread, make ServerThread implement Runnable
4. Create the ServerThread constructor and add the method run.

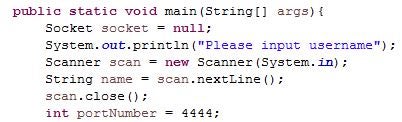
**Step 6: Make the Client Thread Send and Receive Data**

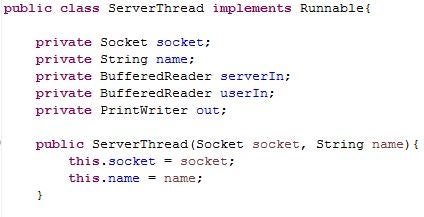




1. In ClientThread make a private BufferedReader in to receive data from clients, and a PrintWriter to write to the client.
2. Create a getter for the PrintWriter
3. Initialize the PrintWriter in run with the socket's output stream, and the BufferedReader with a new InputStreamReader using the socket's input stream
4. Create a while loop in run that checks for any new input and prints the input to all clients using the list of ClientThreads and the getter for the PrintWriter

**Step 7: Make the Server Thread Send and Receive Data**





1. In ServerThread make a private BufferedReader in to receive data from the server, another BufferedReader to receive data from the user, and a PrintWriter to write to the server.
2. Initialize the PrintWriter in run with the socket's output stream, the server's BufferedReader with a new InputStreamReader using the socket's input stream, and the user's BufferedReader with a new InputStreamReader using System.In
3. Create a while loop in run that checks for any new input from the server and prints the input to the console, and checks for any new input from the user and prints that input to the server.

**Step 8: Test the Server and Add Optional Functionality**

After running the server and launching clients, the program should relay all messages from each client to all other clients. This was a small test of what can be done with java sockets.

You can expand on this project by adding functionality to let the server itself relay messages, archiving messages, allowing other forms of input and more to become more familiar with java sockets and networking in general.