Class Client

java.lang.Object
    Client

```
public class Client
extends Object
```

The client will need to enter the server name and port to connect. The client will also select whether to communicate using TCP/IP or UDP.

Field Summary

Fields

Modifier and Type	Field and Description
`private int`	**`bufSize`** the size of the buf for datagram packets
`private ClientGUI`	**`clientGUI`** the object to access the GUI for the client to update the current chat messages
`private boolean`	**`connection`** the client is currently connect
`private BufferedReader`	**`in`** create a BufferedReader to read from the client
`private int`	**`longWait`** the time in milliseconds that the program waits to receive a message during UDP
`private PrintWriter`	**`out`** the object to send messages to the TCP/IP server
`private int`	**`portUDP`** the client's unique port to send UDP messages
`private String`	**`protocol`** Initialize the communication choice
`private String`	**`serverName`** the name of the server the user inputed

`private int`	**`shortWait`** the time in milliseconds that the program waits for server to respond during UDP
`private Socket`	**`socketTCP`** Initialize the client TCP/IP socket
`private DatagramSocket`	**`socketUDP`** Initialize the client UDP socket

Constructor Summary

Constructors

Constructor and Description
`Client``()` Constructor to start the GUI and wait to receive messages

Method Summary

All Methods Static Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
`void`	**`closeSocket`**`()` Closes the current socket.
`void`	**`connectToServer`**`()` This is called when the user submits the server information.
`void`	**`disconnectFromServer`**`()` disconnect from the server, this is called when the user exits
`String`	**`getProtocol`**`()` Get the current communication protocol being used
`private void`	**`hault`**`()` Doesn't allow the client to do anything.
`static void`	**`main`**`(String[] args)` Asks the Server to see a text file
`void`	**`receiveMessages`**`()` Infinitely waits to receive a message using TCP/IP or UDP protocol

`void`	`sendMessage()` Send a message using TCP/IP or UDP protocol
`void`	`setProtocol(String protocol)` Set the current communication protocol being used

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Field Detail

serverName

`private String serverName`

the name of the server the user inputed

portUDP

`private int portUDP`

the client's unique port to send UDP messages

socketTCP

`private Socket socketTCP`

Initialize the client TCP/IP socket

socketUDP

`private DatagramSocket socketUDP`

Initialize the client UDP socket

protocol

`private String protocol`

Initialize the communication choice

clientGUI

```
private ClientGUI clientGUI
```

the object to access the GUI for the client to update the current chat messages

connection

```
private boolean connection
```

the client is currently connect

out

```
private PrintWriter out
```

the object to send messages to the TCP/IP server

in

```
private BufferedReader in
```

create a BufferedReader to read from the client

shortWait

```
private final int shortWait
```

the time in milliseconds that the program waits for server to respond during UDP

See Also:

`Constant Field Values`

longWait

```
private final int longWait
```

the time in milliseconds that the program waits to receive a message during UDP

See Also:

`Constant Field Values`

bufSize

```
private final int bufSize
```

the size of the buf for datagram packets

See Also:

`Constant Field Values`

Constructor Detail

Client

```
public Client()
```

Constructor to start the GUI and wait to receive messages

Method Detail

connectToServer

```
public void connectToServer()
```

This is called when the user submits the server information. It connects the client to the server and waits to receive messages.

receiveMessages

```
public void receiveMessages()
```

Infinitely waits to receive a message using TCP/IP or UDP protocol

disconnectFromServer

```
public void disconnectFromServer()
```

disconnect from the server, this is called when the user exits

sendMessage

```
public void sendMessage()
```

Send a message using TCP/IP or UDP protocol

getProtocol

```
public String getProtocol()
```

Get the current communication protocol being used

Returns:

`the current communication protocol`

setProtocol

```
public void setProtocol(String protocol)
```

Set the current communication protocol being used

Parameters:

`protocol - the current communication protocol (UDP or TCP/IP)`

closeSocket

```
public void closeSocket()
```

Closes the current socket. Is called when the client exits the window.

hault

```
private void hault()
```

Doesn't allow the client to do anything. This is called when the server crashes.

main

```
public static void main(String[] args)
```

Asks the Server to see a text file

Parameters:

`args - Server_name Server_port`