**Color Server and Client Application**

**Overview**

This project demonstrates a basic client-server application using Java and TCP/IP. It allows multiple clients to request random colors from a server, which responds with a randomly selected color and keeps track of the number of requests made by each client. This project is useful for learning the basics of network programming, multithreading, and object serialization in Java.

**Files**

* **ColorServer.java**: Implements the server-side application.
* **ColorClient.java**: Implements the client-side application.

**How It Works**

1. **ColorServer**:
   * Listens for incoming client connections on port 45565.
   * Handles each client connection in a separate thread using the ColorWorker class.
   * Generates a random color in response to client requests and keeps track of the number of requests from each client.
2. **ColorClient**:
   * Connects to the server and requests a color.
   * Sends the client's username and requested color to the server.
   * Receives a randomly selected color from the server along with a message and the total number of requests made.

**Usage**

**Compilation**

sh

Copy code

javac \*.java

**Running the Programs**

To run the server and client on the same machine in different terminal windows:

**On Localhost**

1. **Start the server**:

sh

Copy code

java ColorServer

1. **Start the client** (in separate terminal windows):

sh

Copy code

java ColorClient

**Over the Internet**

1. **Start the server**:

sh

Copy code

java ColorServer

1. **Start the client** (replace 172.16.0.98 with the actual IP address of the server):

sh

Copy code

java ColorClient 172.16.0.98

**Example Usage**

1. **Server Output**:

sh

Copy code

Rohan Dhoyda's Color Server 1.0 starting up, listening at port 45565.

ServerSocket waiting for client to initiate connections...

Connection initiated with Socket[addr=/127.0.0.1,port=61348,localport=45565]

CLIENT REQUEST:

Name of the user Rohan

Color request from client: cyan

connections count: 1

Closing the client socket connection...

Connection initiated with Socket[addr=/127.0.0.1,port=61363,localport=45565]

CLIENT REQUEST:

Name of the user Rohan

Color request from client: blue

connections count: 2

Closing the client socket connection...

1. **Client Output**:

sh

Copy code

To Request a color you need to give your name: Rohan

Hi Rohan

Choose a color you want: cyan

We have successfully connected to the ColorServer at port 45565

We have sent the serialized values to the ColorServer's server socket

RESPONSE RECEIVED:

Thanks Rohan for sending the color cyan

The color sent back is: Soft White

The color count is: 1

Closing the connection to the server.

Choose a color you want: blue

We have successfully connected to the ColorServer at port 45565

We have sent the serialized values to the ColorServer's server socket

RESPONSE RECEIVED:

Thanks Rohan for sending the color blue

The color sent back is: Magenta

The color count is: 2

Closing the connection to the server.

**Notes**

* The server can handle multiple client connections simultaneously using multithreading.
* The communication between client and server is implemented using object serialization.
* The server responds to each client request with a randomly selected color from a predefined list.
* The project demonstrates a connectionless, stateless protocol by terminating the TCP/IP connection after each request is fulfilled.

**Credits**

* [Sending objects over sockets in Java example by Ramesh](https://www.comrevo.com/2019/07/Sending-objects-over-sockets-Java-example-How-to-send-serialized-object-over-network-in-Java.html)
* [Java SocketException by Rollbar](https://rollbar.com/blog/java-socketexception/)
* Hughes, Shoffner, and Winslow for Inet code.