Blocking TCP based sockets in Java

Socket server source code

```
    import java.io.IOException;
    import java.io.ObjectInputStream;
    import java.io.ObjectOutputStream;
    import java.net.ServerSocket;
    import java.net.Socket;

    public class Server {
        ServerSocket providerSocket;
        Socket connection = null;
        ObjectOutputStream out;
        ObjectInputStream in;
        String message;

        Server() {
        }

        void run() {
            try {
                // 1. create a socket server listening to port 8080
                providerSocket = new ServerSocket(8080, 10);
                // 2. waiting for the connection (here we are waiting until
next connection)
                connection = providerSocket.accept();
                // 3. create Input and Output streams
                out = new ObjectOutputStream(connection.getOutputStream());
                in = new ObjectInputStream(connection.getInputStream());
                // 4. socket communication
                do {
                    try {
                        message = (String) in.readObject();
                        System.out.println("client>" + message);
                        if (message.equals("bye")) {
                            sendMessage("bye");
                        }
                    } catch (ClassNotFoundException classnot) {
                        System.err.println("Data received in unknown
format");
                    }
                } while (!message.equals("bye"));
            } catch (IOException ioException) {
                ioException.printStackTrace();
            } finally {
                // 4: close connection
                try {
                    in.close();
                    out.close();
```

```
                    providerSocket.close();
                } catch (IOException ioException) {
                    ioException.printStackTrace();
                }
            }
        }

        void sendMessage(String msg) {
            try {
                out.writeObject(msg);
                out.flush();
                System.out.println("server>" + msg);
            } catch (IOException ioException) {
                ioException.printStackTrace();
            }
        }

        public static void main(String args[]) {
            Server server = new Server();
            while (true) {
                server.run();
            }
        }
    }
```

Socket client source

```
    import java.io.IOException;
    import java.io.ObjectInputStream;
    import java.io.ObjectOutputStream;
    import java.net.Socket;
    import java.net.UnknownHostException;

    public class Client {
        Socket requestSocket;
        ObjectOutputStream out;
        ObjectInputStream in;
        String message;

        Client() {
        }

        void run() {
            try {
                // 1. try to connect to the socket: localhost:8080
                requestSocket = new Socket("localhost", 8080);
                // 2. Input and Output streams
                out = new
ObjectOutputStream(requestSocket.getOutputStream());
```

```
                in = new ObjectInputStream(requestSocket.getInputStream());
                // 3: communications
                do {
                    try {
                        sendMessage("Hello server");
                        sendMessage("bye");
                        message = (String) in.readObject();
                    } catch (Exception e) {
                        System.err.println("data received in unknown
format");
                    }
                } while (!message.equals("bye"));
            } catch (UnknownHostException unknownHost) {
                System.err.println("You are trying to connect to an unknown
host!");
            } catch (IOException ioException) {
                ioException.printStackTrace();
            } finally {
                // 4: close connection
                try {
                    in.close();
                    out.close();
                    requestSocket.close();
                } catch (IOException ioException) {
                    ioException.printStackTrace();
                }
            }
        }

        void sendMessage(String msg) {
            try {
                out.writeObject(msg);
                out.flush();
                System.out.println("client>" + msg);
            } catch (IOException ioException) {
                ioException.printStackTrace();
            }
        }

        public static void main(String args[]) {
            Client client = new Client();
            client.run();
        }
    }
```

Non-blocking TCP based sockets in Java

The following Agent sends a message and waits for a response on port 8080, also with UDP. Older versions of the Eclipse IDE, the text you type on the console can be sent by pressing ctrl+z

```
  package org.ait;

  import java.io.BufferedReader;
  import java.io.InputStreamReader;
  import java.net.DatagramPacket;
  import java.net.DatagramSocket;
  import java.net.InetAddress;

  public class UDPClient {
    public static void main(String args[]) throws Exception {
        BufferedReader inFromUser = new BufferedReader(new
InputStreamReader(System.in));
        DatagramSocket clientSocket = new DatagramSocket();
        InetAddress IPAddress = InetAddress.getByName("localhost");

        byte[] sendData = new byte[1024];
        byte[] receiveData = new byte[1024];

        String sentence = inFromUser.readLine();
        sendData = sentence.getBytes();

        DatagramPacket sendPacket = new DatagramPacket(sendData,
sendData.length, IPAddress, 8080);
        clientSocket.send(sendPacket);

        DatagramPacket receivePacket = new DatagramPacket(receiveData,
receiveData.length);
        clientSocket.receive(receivePacket);
        String modifiedSentence = new String(receivePacket.getData());

        System.out.println("converted:" + modifiedSentence);
        clientSocket.close();
    }
    }
```

The UDP server waits for the agents messages on port 8080 and converts them to uppercase letters and sends them back to the client UDP socket.

```
  package org.ait;

  import java.net.DatagramPacket;
  import java.net.DatagramSocket;
  import java.net.InetAddress;

  public class UDPServer {
    public static void main(String args[]) throws Exception {

        DatagramSocket serverSocket = new DatagramSocket(8080);

        byte[] bytesReceived = new byte[1024];
```

```
        byte[] bytesSent = new byte[1024];


        DatagramPacket receivePacket = new DatagramPacket(bytesReceived,
bytesReceived.length);
        // here we are waiting for the packets
        serverSocket.receive(receivePacket);

        String textMessage = new String(receivePacket.getData());

        System.out.println("I got: " + textMessage);

        InetAddress IPAddress = receivePacket.getAddress();
        int port = receivePacket.getPort();

        String upperCaseText = textMessage.toUpperCase();
        bytesSent = upperCaseText.getBytes();

        // send back
        DatagramPacket sendPacket = new DatagramPacket(bytesSent,
bytesSent.length, IPAddress, port);
        serverSocket.send(sendPacket);
        serverSocket.close();

    }
  }
```