Java and Networking

Two or more Java applications can communicate via network by using stream of byte of data as a message. This type of communication is known as "Server and Client" application. The data being transferred can be communicate with two types of network protocols.

- TCP (Transmission Control Protocol) This type of protocol requires connection to be requested by a client to the server which must be accepted before the client can begin transmitting data.
 - Connection Oriented Protocol
 - Secured but slow
 - Server has to accept connection from client
 - Package will be arrived in the order
 - Example: A web server connects to credit card server for a transaction.
- UDP (User Datagram Protocol) — This type of protocol does not require the server to accept a connection from any client. The client can just begin transmitting data to the server without having to make a request.
 - Connectionless Protocol
 - Unsecured but fast
 - Server doesn't need to accept connection from client
 - Package will be sent in random order and never know whether it is going to arrive at the destination
 - o Example: A web browser connects to YouTube for a video.

NOTE: TCP will be used as an example for this class.

Sockets

A socket is an endpoint of the connection between the server and client. A Java server application that uses TCP protocol must provide port number and the maximum number of connections simultaneously. A Java client application then must use the same protocol to connect to server by including the IP address or the host name of server and the port number to make a request of connection. If it is accepted then, the transmission of data can begin.

NOTE: Refer to the comments in the sample code for details of Java objects used to implement server & client programs.

Java Example: One-way communication (From client to server)

Server: (Must be run before the client to wait for a connection)

```
1 /*
 2
    Using TCP protocal (Transfer Control Protocal)
 3
   One-Way Communication
 4 */
 5 // import objects for server
 6 import java.net.ServerSocket;
 7 import java.net.Socket;
 8 import java.io.ObjectInputStream;
 9 import java.io.IOException;
10
11 public class TCPServer {
12
     private ServerSocket serverSocket; // declare abject for server's socket information
13
     private Socket socket; // declare an actual socket
14
    private ObjectInputStream input; // declare an output stream to send a mdssage
15
16
     public TCPServer() {
17
       System.out.println("Server is running...");
18
        try {
19
            // initialize socket information to receive message from the Client on port 1098
20
           // a maximum number of 500 clients connected silmultaneously
21
           serverSocket = new ServerSocket(1098, 500);
22
           while(true) { // use a loop to keep server running
23
             socket = serverSocket.accept(); // accept connection from client
24
              input = new ObjectInputStream(socket.getInputStream()); // receive output stream object
25
              String message = (String) input.readObject(); // convert stream byte to String
              System.out.println("Client says: " + message); // display the message received from client
26
27
            }
28
        }
29
        catch(IOException ioe) { ioe.printStackTrace(); }
       catch(ClassNotFoundException cnfe) { cnfe.printStackTrace(); }
     public static void main(String [] args) {
33
       new TCPServer();
34
      }
35 }
 Interactions
               Console
                         Compiler Output
Welcome to DrJava. Working directory
> run TCPServer
Server is running...
```

Server is running and waiting for a communication from a client.

Client:

```
1 /*
    Using TCP protocal (Transfer Control Protocal)
 3
    One-Way Communication
 4 */
 5 // import objects for client
 6 import java.net.Socket;
 7
   import java.io.ObjectOutputStream;
8 import java.net.InetAddress;
9 import java.net.UnknownHostException;
10 import java.io.IOException;
11 import java.util.Scanner;
12
13 public class TCPClient {
14
      private Socket socket; // declare an actual socket
15
      private ObjectOutputStream output; // declare an output stream to send a mdssage
16
17
      public TCPClient() {
18
        try {
19
           while(true) { // use a loop to keep client running
20
             // initialize socket to send a message to server running on port 1098
             socket = new Socket(InetAddress.getByName("localhost"), 1098);
22
             // initializa output stream object
23
             output = new ObjectOutputStream(socket.getOutputStream());
24
25
             Scanner scan = new Scanner(System.in);
26
             System.out.print("Client says:");
27
             String message = scan.nextLine(); // Scanner to type in a message sent to the client
28
29
             // write buffered output bytes and flush through to the underlying stream
30
             output.writeObject(message);
31
             output.flush();
32
             System.out.println("Message sent!!!");
33
34
        catch(UnknownHostException uhe) { uhe.printStackTrace(); }
36
        catch(IOException ioe) { ioe.printStackTrace(); }
37
38
39
      public static void main(String [] args) {
40
        new TCPClient();
41
      }
42 }
                    Compiler Output
 Interactions
            Console
Welcome to DrJava. Working directory
                                          Interactions
                                                     Console
                                                             Compiler Output
> run TCPClient
Client says: Hello Server
                                        Welcome to DrJava. Working directory
                                        > run TCPServer
Message sent!!!
                                         Server is running...
Client says:
                                         Client says: Hello Server
```

Message is sent from the client to the server.

Server: Must be run first

```
2
    Using TCP protocal (Transfer Control Protocal)
 3
    Two-Way Communication
 4 */
 5
   // import objects for server
   import java.net.ServerSocket;
    import java.net.Socket;
   import java.io.ObjectInputStream;
 9 import java.io.IOException;
10
11 // objects for creating client
12 import java.io.ObjectOutputStream;
13 import java.net.InetAddress;
14 import java.util.Scanner;
15
16 public class TCPServer {
17
      private ServerSocket serverSocket; // declare abject for server's socket information
18
      private Socket socket; // declare an actual socket
19
      private ObjectInputStream input; // declare an output stream to received a mdssage
20
      private ObjectOutputStream output; // declare an output stream to send a mdssage
21
22
      public TCPServer() {
23
        System.out.println("Server is running...");
24
        Scanner scanner = new Scanner(System.in);
25
        try {
26
            // initialize socket information to receive message from the client on port 1098
            // a maximum number of 500 clients connected silmultaneously
27
28
            serverSocket = new ServerSocket(1098, 500);
29
            while(true) { // use a loop to keep server running
30
              socket = serverSocket.accept(); // accept connection from client
31
              input = new ObjectInputStream(socket.getInputStream()); // receive output stream object
32
              String message = (String) input.readObject(); // convert stream byte to String
33
              System.out.println("Client says: " + message); // display the message received from client
34
35
              // initialize socket to send a message to client running on port 1097
36
              socket = new Socket(InetAddress.getByName("localhost"), 1097);
37
              // initialize output stream object
38
              output = new ObjectOutputStream(socket.getOutputStream());
39
              System.out.print("Server Say:");
40
              String message2 = scanner.nextLine();
41
42
               // write buffered output bytes and flush through to the underlying stream
43
              output.writeObject(message2);
44
              output.flush();
45
           }
46
        1
47
        catch(IOException ioe) { ioe.printStackTrace(); }
48
        catch(ClassNotFoundException cnfe) { cnfe.printStackTrace(); }
49
50
      public static void main(String [] args) {
51
        new TCPServer();
52
53 }
```

Client: This will make a request to server and begin a chat between the two applications.

```
1 /*
2 Using TCP protocal (Transfer Control Protocal)
 3
    Two-Way Communication
 4 */
 5 // objects for creating client
 6 import java.io.ObjectOutputStream;
 7 import java.net.InetAddress;
 8 import java.util.Scanner;
10 // objects for creating server
11 import java.net.ServerSocket;
12 import java.net.Socket;
13 import java.io.ObjectInputStream;
14 import java.io.IOException;
15
16 public class TCPClient {
17
18
      private Socket socket; // declare an actual socket
19
      private ObjectOutputStream output; // declare an output stream to send a mdssage
20
21
      private ServerSocket serverSocket; // declare abject for server's socket information
22
      private ObjectInputStream input; // declare an output stream to send a mdssage
23
24
     public TCPClient() {
25
        System.out.println("Client is running...");
26
        Scanner scanner = new Scanner(System.in);
27
28
         // initialize another socket information to receive message from the server on port 1097
29
         // a maximum number of 500 clients connected silmultaneously
30
          serverSocket = new ServerSocket(1097, 500);
31
         while(true) { // use a loop to keep client running
32
             // initialize socket to send a message to server running on port 1098
33
             socket = new Socket(InetAddress.getByName("localhost"), 1098);
34
             // initializa output stream object
35
             output = new ObjectOutputStream(socket.getOutputStream());
36
             System.out.print("Client Says:");
37
             String message = scanner.nextLine();
38
39
             // write buffered output bytes and flush through to the underlying stream
40
             output.writeObject(message);
41
             output.flush();
42
             // System.out.println("Message sent!!!");
43
44
             socket = serverSocket.accept(); // accept connection from server
45
             input = new ObjectInputStream(socket.getInputStream()); // receive output stream object
46
             String message2 = (String) input.readObject(); // convert stream byte to String
47
             System.out.println("Server says: " + message2); // display the message received from client
48
         }
49
50
        catch(IOException ioe) {
51
         ioe.printStackTrace();
52
53
        catch(ClassNotFoundException cnfe) { // do not need to import because it's part of java.lang
54
          cnfe.printStackTrace();
55
        }
56
    }
57
58
    public static void main(String [] args) {
    new TCPClient();
60
61 }
```