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
  - Example: A web browser connects to YouTube for a video.

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

Sockets

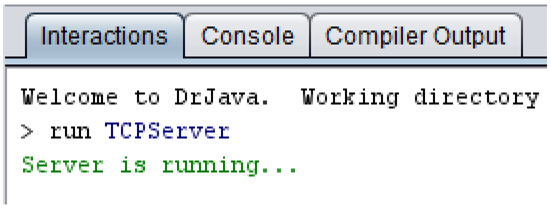
A socket is an endpoint of the connection between the server and client. A Java server application that uses TCP protocol must provide port number and the maximum number of connections simultaneously. A Java client application then must use the same protocol to connect to server by including the IP address 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)

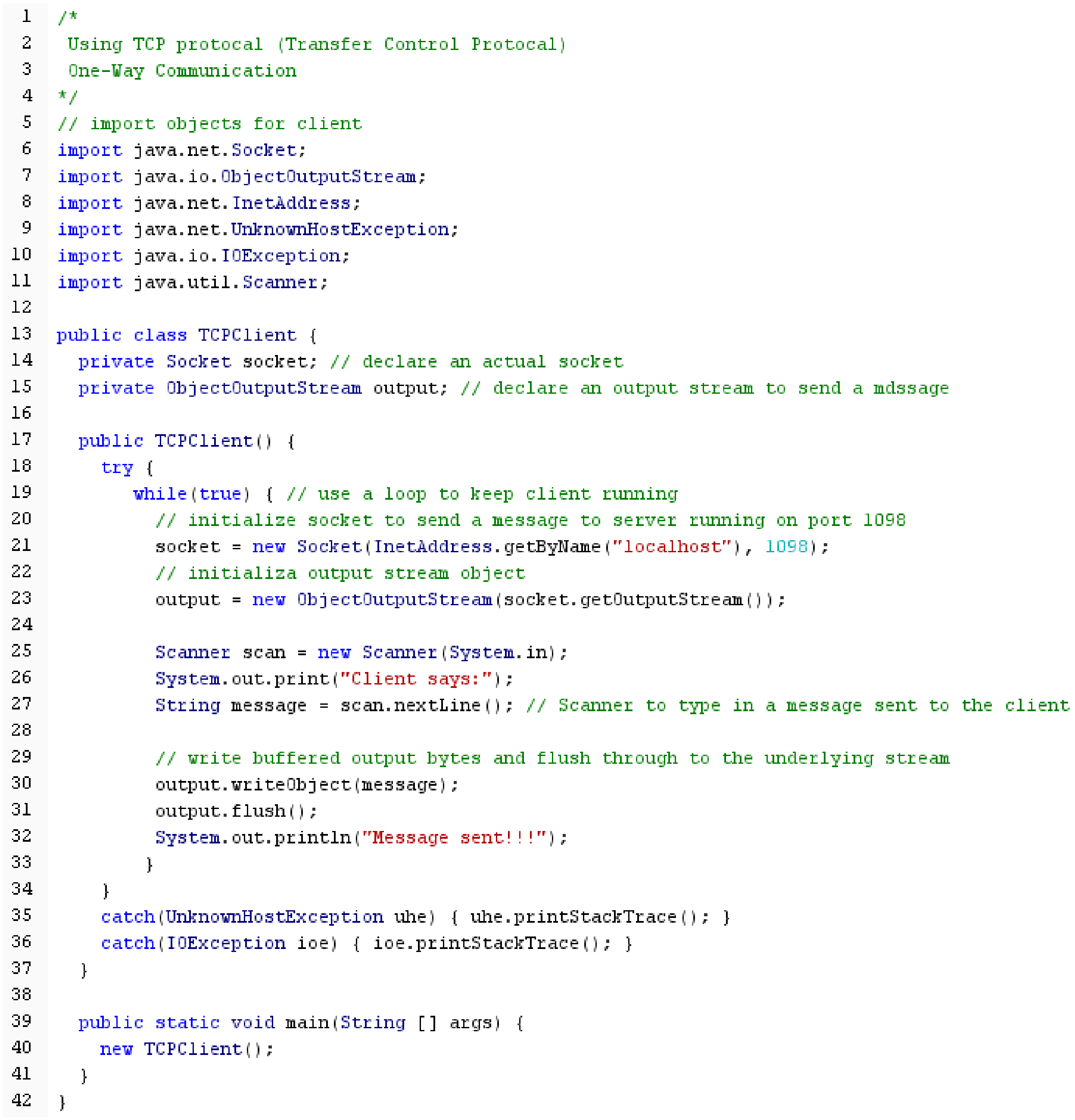
```java
/*
 Using TCP protocal (Transfer Control Protocal)
 One-Way Communication
*/
// import objects for server
import java.net.ServerSocket;
import java.net.Socket;
import java.io.ObjectInputStream;
import java.io.IOException;

public class TCPServer {
  private ServerSocket serverSocket; // declare abject for server's socket information
  private Socket socket; // declare an actual socket
  private ObjectInputStream input; // declare an output stream to send a mdssage

  public TCPServer() {
    System.out.println("Server is running...");
    try {
        // initialize socket information to receive message from the Client on port 1098
        // & maximum number of 500 clients connected silmultaneously
        serverSocket = new ServerSocket(1098, 500);
        while(true) { // use a loop to keep server running
          socket = serverSocket.accept(); // accept connection from client
          input = new ObjectInputStream(socket.getInputStream()); // receive output stream object
          String message = (String) input.readObject(); // convert stream byte to String
          System.out.println("Client says: " + message); // display the message received from client
        }
    }
    catch(IOException ioe) { ioe.printStackTrace(); }
    catch(ClassNotFoundException cnfe) { cnfe.printStackTrace(); }
  }
  public static void main(String [] args) {
    new TCPServer();
  }
}
```



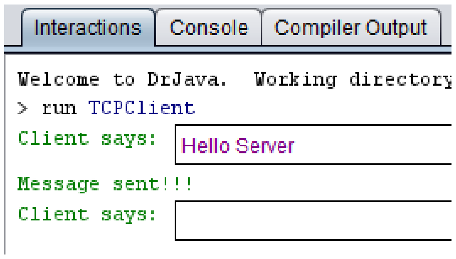
```
Welcome to DrJava.  Working directory
> run TCPServer
Server is running...
```

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

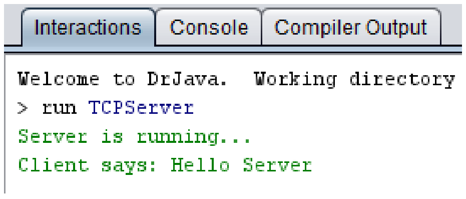
Client:



```java
/*
 Using TCP protocal (Transfer Control Protocal)
 One-Way Communication
*/
// import objects for client
import java.net.Socket;
import java.io.ObjectOutputStream;
import java.net.InetAddress;
import java.net.UnknownHostException;
import java.io.IOException;
import java.util.Scanner;

public class TCPClient {
  private Socket socket; // declare an actual socket
  private ObjectOutputStream output; // declare an output stream to send a mdssage

  public TCPClient() {
    try {
       while(true) { // use a loop to keep client running
         // initialize socket to send a message to server running on port 1098
         socket = new Socket(InetAddress.getByName("localhost"), 1098);
         // initializa output stream object
         output = new ObjectOutputStream(socket.getOutputStream());

         Scanner scan = new Scanner(System.in);
         System.out.print("Client says:");
         String message = scan.nextLine(); // Scanner to type in a message sent to the client

         // write buffered output bytes and flush through to the underlying stream
         output.writeObject(message);
         output.flush();
         System.out.println("Message sent!!!");
        }
    }
    catch(UnknownHostException uhe) { uhe.printStackTrace(); }
    catch(IOException ioe) { ioe.printStackTrace(); }
  }

  public static void main(String [] args) {
    new TCPClient();
  }
}
```



```
Welcome to DrJava.  Working directory
> run TCPClient
Client says: Hello Server
Message sent!!!
Client says:
```



```
Welcome to DrJava.  Working directory
> run TCPServer
Server is running...
Client says: Hello Server
```

Message is sent from the client to the server.

Example: Two ways communication as a simple chat program

Server: Must be run first

```java
/*
 Using TCP protocal (Transfer Control Protocal)
 Two-Way Communication
*/
// import objects for server
import java.net.ServerSocket;
import java.net.Socket;
import java.io.ObjectInputStream;
import java.io.IOException;

// objects for creating client
import java.io.ObjectOutputStream;
import java.net.InetAddress;
import java.util.Scanner;

public class TCPServer {
  private ServerSocket serverSocket; // declare abject for server's socket information
  private Socket socket; // declare an actual socket
  private ObjectInputStream input; // declare an output stream to received a mdssage
  private ObjectOutputStream output; // declare an output stream to send a mdssage

  public TCPServer() {
    System.out.println("Server is running...");
    Scanner scanner = new Scanner(System.in);
    try {
        // initialize socket information to receive message from the client on port 1098
        // & maximum number of 500 clients connected silmultaneously
        serverSocket = new ServerSocket(1098, 500);
        while(true) { // use a loop to keep server running
          socket = serverSocket.accept(); // accept connection from client
          input = new ObjectInputStream(socket.getInputStream()); // receive output stream object
          String message = (String) input.readObject(); // convert stream byte to String
          System.out.println("Client says: " + message); // display the message received from client

          // initialize socket to send a message to client running on port 1097
          socket = new Socket(InetAddress.getByName("localhost"), 1097);
          // initialize output stream object
          output = new ObjectOutputStream(socket.getOutputStream());
          System.out.print("Server Say:");
          String message2 = scanner.nextLine();

           // write buffered output bytes and flush through to the underlying stream
          output.writeObject(message2);
          output.flush();
        }
    }
    catch(IOException ioe) { ioe.printStackTrace(); }
    catch(ClassNotFoundException cnfe) { cnfe.printStackTrace(); }
  }
  public static void main(String [] args) {
    new TCPServer();
  }
}
```

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

```java
/*
 Using TCP protocal (Transfer Control Protocal)
 Two-Way Communication
*/
// objects for creating client
import java.io.ObjectOutputStream;
import java.net.InetAddress;
import java.util.Scanner;

// objects for creating server
import java.net.ServerSocket;
import java.net.Socket;
import java.io.ObjectInputStream;
import java.io.IOException;

public class TCPClient {

  private Socket socket; // declare an actual socket
  private ObjectOutputStream output; // declare an output stream to send a mdssage

  private ServerSocket serverSocket; // declare abject for server's socket information
  private ObjectInputStream input; // declare an output stream to send a mdssage

  public TCPClient() {
    System.out.println("Client is running...");
    Scanner scanner = new Scanner(System.in);
    try {
      // initialize another socket information to receive message from the server on port 1097
      // & maximum number of 500 clients connected silmultaneously
      serverSocket = new ServerSocket(1097, 500);
      while(true) { // use a loop to keep client running
          // initialize socket to send a message to server running on port 1098
         socket = new Socket(InetAddress.getByName("localhost"), 1098);
         // initializa output stream object
         output = new ObjectOutputStream(socket.getOutputStream());
         System.out.print("Client Says:");
         String message = scanner.nextLine();

         // write buffered output bytes and flush through to the underlying stream
         output.writeObject(message);
         output.flush();
         // System.out.println("Message sent!!!");

         socket = serverSocket.accept(); // accept connection from server
         input = new ObjectInputStream(socket.getInputStream()); // receive output stream object
         String message2 = (String) input.readObject(); // convert stream byte to String
         System.out.println("Server says: " + message2); // display the message received from client
      }
    }
    catch(IOException ioe) {
      ioe.printStackTrace();
    }
    catch(ClassNotFoundException cnfe) { // do not need to import because it's part of java.lang
      cnfe.printStackTrace();
    }
 }

 public static void main(String [] args) {
  new TCPClient();
 }
}
```