

Esmaeilsadeghi

## فهرت

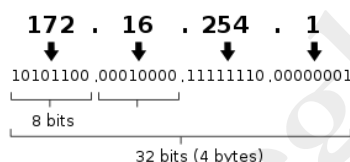
۲	سوکت.....
۲	آدرس آی پی.....
۲	پورت.....
۳	پروتکل.....
۳	برنامه نویسی سوکت.....
۴	انواع سوکت ها.....
۴	معماری سوکت.....
۶	برنامه نویسی سوکت در جاوا.....
۷	کلاس سوکت.....
۷	کلاس سرور سوکت.....
۱۲	سوالات.....

## سوکت

سوکت شامل کانال ارتباطی است که مبادله اطلاعات از طریق آن انجام می‌شود. همچنین سوکت این امکان را فراهم می‌کند تا داده‌ها به صورت دوطرفه در هر لحظه ارسال و دریافت شوند. به بیانی ساده تر، می‌توان گفت که سوکت به ترکیب یک آدرس ماشین (آی پی<sup>۱</sup>) و یک شماره درگاه (پورت<sup>۲</sup>) گفته می‌شود. که در ادامه به بررسی این مفاهیم می‌پردازیم.

## آدرس آی پی

هر کامپیوتر و ماشینی که به شبکه متصل است، یک آدرس آی پی یکتا دارد، به عبارت دیگر، هر کامپیوتر به ازای هر کارت شبکه‌اش یک آدرس آی پی دارد که از طریق آن ماشین‌های دیگر در شبکه می‌توانند به آن متصل شوند. آدرس آی پی دارای دو نوع آی پی نسخه ۴ و آی پی نسخه ۶ (IPv۴ و IPv۶) می‌باشد که هر یک نشانی آی پی را به روش متفاوتی ارائه می‌نمایند. نشانی آی پی نسخه ۴ به شکل زیر می‌باشد:



همچنین آی پی آدرس نسخه ۶ شامل ترکیب ۱۲۸ بیتی است این ترکیب اعداد دارای ۸ بخش ۱۶ بیتی است که علاوه بر نقطه از دو نقطه استفاده می‌کند این ترکیب از هگزادسیمال (عدد در مینا ۱۶) استفاده میشود، هر بخش ۱۶ بیتی محدوده‌ای از 0000 تا FFFF را در ترکیب خود جای می‌دهد. مثلاً با استفاده از آدرس زیر می‌توانید مشخصات و نوع آی پی سیستم خود را مشخص کنید:

<https://ipnumberia.com/>

## پورت

پورت یک مفهوم منطقی است که به کمک آن می‌توان به طور همزمان با چندین ماشین دیگر ارتباط برقرار نمود. پورت‌ها به دو گروه رزرو شده و غیر رزرو شده تقسیم می‌شوند. پورت‌های رزرو شده (پورت‌های بین ۱ تا ۱۰۲۴)، برای کاربردهای استاندارد مورد استفاده قرار می‌گیرند. مثلاً: در برنامه‌های سرور/کلاينت<sup>۳</sup>، از پورت‌های غیر رزرو شده (سایر پورت‌ها) که آزاد باشند و مورد استفاده‌ی سایر برنامه‌ها نباشند، می‌توان جهت برقراری ارتباطات مورد نیاز، استفاده نمود. یعنی می‌تواند به ازای هر پورت، با یک برنامه ارتباط برقرار کند.

<sup>1</sup> Internet Protocol (IP)

<sup>2</sup> Port

<sup>3</sup> Server/Client

## پروتکل

قرارداد یا پروتکل<sup>۴</sup>، قوانین و قراردادهایی هستند که تعیین می‌کنند، برنامه‌ها و ماشین‌های مختلف چگونه با هم ارتباط برقرار کنند (با هم صحبت کنند). در واقع، پروتکل‌های شبکه، ماشین‌ها را قادر می‌سازند تا از طریق یک زبان مشترک (مثل زبان انگلیسی، عربی، فارسی) با یکدیگر صحبت کرده و تبادل اطلاعات کنند. به عنوان مثال HTTP و FTP و SMTP و POP. همچنین سوکت یه دستگاه ارتباطی مثل تلفن است. شما با تلفن زنگ می‌زنید به طرف مقابل و اون گوشی رو برمی‌داره و حالا یک کانال ارتباطی بین شما وجود داره که میتونید از طریق اون صدای همدیگر رو بشنوید. این یعنی سوکت. اما فقط توانایی تبادل صدا کافی نیست، بلکه نیاز هست طرفین به زبان مشترکی صحبت کنن تا حرف همدیگر رو متوجه بشید. این زبان مشترک همون پروتکل‌ها هستند.

شبکه دارای چند نوع پروتکل است که از بین آن‌ها TCP<sup>۵</sup> و UDP<sup>۶</sup> بیش‌ترین مورد استفاده را دارند. پروتکل TCP به معنای پروتکل کنترل انتقال است و در جایی استفاده می‌شود که لازم باشد انتقال داده‌ها با امنیت بالا، به ترتیب و بی‌عیب و نقص انجام گیرد. در این پروتکل خطاها بررسی شده و اگر داده‌ای دریافت نشده باشد، دوباره ارسال می‌گردد. مانند دانلود فایل در اینترنت. پروتکل UDP به معنای پروتکل بسته‌داده‌ی کاربر است. این نوع پروتکل برای کاربردهایی استفاده می‌شود که نیاز به اتصال ندارند و داده‌های از دست رفته، دوباره ارسال نمی‌شوند. مانند پخش فایل تصویری در اینترنت.



## برنامه نویسی سوکت

سوکت امکان ارتباط بین دو فرایند مختلف را در دستگاه‌های مشابه یا متفاوت فراهم می‌کند. به عبارت دیگر، روشی برای صحبت کردن با کامپیوترهای دیگر با استفاده از توصیف گرهای استاندارد فایل است. مثلاً در یونیکس، هر عمل ورودی و خروجی با نوشتن یا خواندن یک توصیف گر فایل انجام می‌شود. یک توصیف گر فایل می‌تواند یک عدد صحیح یا یک فایل باز باشد و می‌تواند یک اتصال شبکه، یک فایل متنی، یک ترمینال یا چیز دیگری باشد. برای یک برنامه نویس، سوکت مانند یک توصیف گر<sup>۷</sup> فایل سطح پایین رفتار می‌کند و به نظر می‌رسد. این به این دلیل است که دستوراتی مانند خواندن<sup>۸</sup> و نوشتن<sup>۹</sup> با سوکت‌ها به همان شکلی که آنها با فایل‌ها و پایپ‌ها کار می‌کنند است. برنامه نویسی سوکت راهی برای اتصال دو گره در یک شبکه برای برقراری ارتباط با

<sup>4</sup> Protocol

<sup>5</sup> Transmission Control Protocol

<sup>6</sup> User datagram protocol

<sup>7</sup> Descriptor

<sup>8</sup> Read

<sup>9</sup> Write

یکدیگر است. یک سوکت در یک درگاه خاص در ای پی گوش می دهد، تازمانی که سوکت دیگر برای ایجاد اتصال به دیگری می رسد. سرور تازمانی که کلاینت به سرور می رسد، سوکت شنونده را تشکیل می دهد.

## انواع سوکت‌ها

سوکت	توضیحات
استریم <sup>۱۰</sup>	اساس کار این سوکت‌ها بر پایه‌ی پروتکل TCP طراحی شده و با آن کار می‌کنند. در این سوکت‌ها باید قبل از جابه‌جایی اطلاعات، یک اتصال امن و قدرتمند ایجاد شده تا داده‌ها با نظم و دقت، ارسال و دریافت شوند. در ارتباطات با پروتکل‌های HTTP، FTP و SMTP از این نوع سوکت‌ها استفاده می‌شود.
دیتاگرام <sup>۱۱</sup>	این نوع سوکت‌ها براساس پروتکل UDP کار می‌کند و هیچ اتصالی از قبل برای جابه‌جایی داده‌ها لازم ندارد. در این سوکت‌ها، انتقال صحیح و کامل داده‌ها اهمیت ندارد و هیچ وقت رسیدن داده به مقصد بررسی نمی‌شود. مهم‌ترین دلیل استفاده از این سوکت‌ها، سرعت بالای انتقال داده‌هاست و بیش‌تر در انتقال صوت و تصویر استفاده می‌شود.

## معماری سوکت

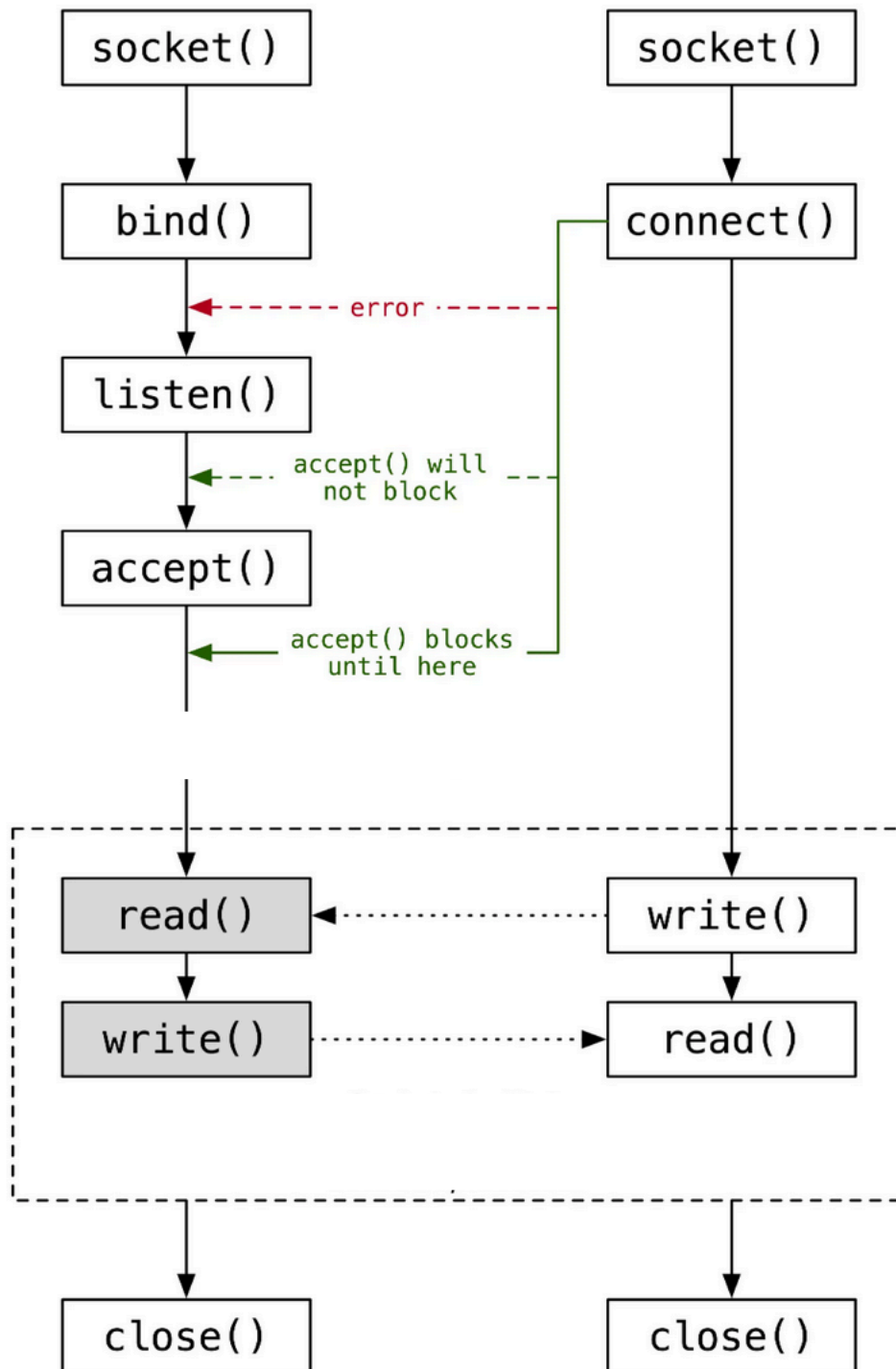
سوکت‌ها معمولاً برای تعامل کلاینت و سرور استفاده می‌شوند. پیکربندی سیستم معمولی سرور را روی یک دستگاه قرار می‌دهد و کلاینت‌ها در دستگاه‌های دیگر قرار دارند. کلاینت‌ها به سرور وصل می‌شوند، اطلاعات را تبادل می‌کنند و سپس قطع می‌شوند. یک سوکت جریان معمولی از وقایع را دارد. در یک مدل سرویس گیرنده به سرور متصل، سوکت در فرآیند سرور منتظر درخواست‌های کلاینت می‌باشد. برای انجام این کار، سرور ابتدا آدرسی را ایجاد می‌کند که می‌تواند از آنها برای پیدا کردن سرور استفاده کنند. هنگامی که آدرس ایجاد شد، سرور منتظر است تا سرویس دهندگان سرویس را درخواست کنند. تبادل داده کلاینت-سرور هنگامی انجام می‌شود که کلاینت از طریق سوکت به سرور متصل شود. سرور درخواست کلاینت را انجام داده و پاسخ را به کلاینت ارسال می‌کند. شکل زیر جریان معمولی وقایع برای یک جلسه سوکت اتصال گرا را نشان می‌دهد. توضیحی درباره هر رویداد از شکل زیر پیروی می‌کند.

<sup>10</sup> Stream

<sup>11</sup> Datagram

Server

Client



جریان معمولی برای یک سوکت اتصال گرا به شرح زیر می باشد:

۱. socket() یک نقطه پایانی برای ارتباطات ایجاد می کند.
۲. وقتی یک برنامه دارای توصیف کننده سوکت است، می تواند یک نام منحصر به فرد را به سوکت وصل کند. سرورها باید نامی را برای دسترسی به شبکه پیوند دهند.
۳. listen() نشان دهنده تمایل به پذیرش درخواست های اتصال کلاینت است. هنگامی که یک listen() برای یک سوکت صادر می شود، آن سوکت نمی تواند درخواست های اتصال را به طور فعال آغاز کند. listen() پس از اختصاص سوکت با socket() صادر می شود و bind() اتصال به یک نام به سوکت صادر می شود. قبل از انتشار accept() باید listen() صادر شود.
۴. برنامه کلاینت برای ایجاد اتصال به سرور از connect() در یک سوکت جریان استفاده می کند.
۵. برنامه سرور برای پذیرش درخواست اتصال کلاینت از accept() استفاده می کند. قبل از اینکه API را بپذیرد، سرور باید API های bind() و listen() را بطور موفقیت آمیز منتشر کند.
۶. هنگامی که اتصال بین سوکت های جریان (بین کلاینت و سرور) برقرار شد، می توانید از هر کدام از API های انتقال داده API سوکت استفاده کنید. کلاینت ها و سرورها API های زیادی برای انتقال داده دارند که از آن می توان انتخاب کرد، مانند send(), recv(), write(), read() و موارد دیگر. هنگامی که یک سرور یا کلاینت می خواهد عملیات را متوقف کند، close() را برای انتشار منابع سیستم به دست آمده توسط سوکت صادر می کند.

## برنامه نویسی سوکت در جاوا

برنامه نویسی سوکت جاوا برای ارتباط بین برنامه های کاربردی در حال اجرا استفاده می شود. برنامه نویسی سوکت جاوا می تواند اتصال گرا یا بدون اتصال باشد. ماژول سوکت جاوا واسط API سوکت های Berkeley را فراهم می کند. توابع و متد های اصلی API سوکت در این ماژول عبارتند از:

- Socket
- ServerSocket
- DatagramSocket
- DatagramPacket

مشتری<sup>۱۲</sup> در برنامه نویسی سوکت باید دو اطلاعات را بداند:

۱. IP Address
۲. Port number.

در اینجا، ما قصد داریم ارتباط یک طرفه مشتری و سرور برقرار کنیم. در این اپلیکیشن، کلاینت پیامی را به سرور ارسال می کند، سرور پیام را می خواند و آن را چاپ می کند. در اینجا از دو کلاس استفاده می شود: Socket و ServerSocket. کلاس Socket برای ارتباط کلاینت و سرور استفاده می شود. از طریق این کلاس می توانیم پیام بخوانیم و بنویسیم. کلاس ServerSocket در سمت سرور استفاده می شود. متد accept() کلاس ServerSocket کنسول را تا زمانی که کلاینت متصل شود مسدود می کند. پس از اتصال موفقیت آمیز مشتری، نمونه سوکت را در سمت سرور برمی گرداند

---

<sup>12</sup> Client

## کلاس سوکت

یک سوکت به سادگی یک نقطه پایانی برای ارتباطات بین ماشین ها است. کلاس Socket می تواند برای ایجاد یک سوکت استفاده شود. این کلاس شامل متدهای زیر می باشد:

متد	توضیحات
<code>getInputStream()</code>	جریان ورودی متصل به این سوکت را برمی گرداند.
<code>getOutputStream()</code>	جریان خروجی متصل به این سوکت را برمی گرداند.
<code>close()</code>	این سوکت را می بندد

## کلاس سرور سوکت

کلاس ServerSocket می تواند برای ایجاد سوکت سرور استفاده شود. این شی برای برقراری ارتباط با مشتریان استفاده می شود. این کلاس شامل متدهای زیر می باشد:

متد	توضیحات
<code>accept()</code>	سوکت را برمی گرداند و بین سرور و کلاینت ارتباط برقرار می کند.
<code>close()</code>	سوکت سرور را می بندد.

## نمونه ای از برنامه نویسی سوکت جاوا

— ایجاد سرور:

برای ایجاد برنامه سرور، باید نمونه کلاس ServerSocket را ایجاد کنیم. در اینجا، ما از شماره پورت 6666 برای ارتباط بین مشتری و سرور استفاده می کنیم. همچنین می توانید هر شماره پورت دیگری را انتخاب کنید. متد `accept` منتظر مشتری می ماند. اگر کلاینت ها با شماره پورت داده شده متصل شوند، نمونه ای از Socket را برمی گرداند.

```
ServerSocket serverSocket = new ServerSocket(6666);
```

```
Socket socket = serverSocket.accept(); // ماند و منتظر مشتری می ماند
```



– ایجاد مشتری:

برای ایجاد اپلیکیشن کلاینت، باید نمونه کلاس Socket را ایجاد کنیم. در اینجا، باید آدرس IP یا نام میزبان سرور و یک شماره پورت را ارسال کنیم. در اینجا، ما از "localhost" استفاده می کنیم زیرا سرور ما روی همان سیستم اجرا می شود.

```
Socket socket =new Socket("localhost",6666);
```

بیایید یک برنامه نویسی سوکت جاوا را ببینیم که در آن کلاینت متنی را ارسال می کند و سرور آن را دریافت و چاپ می کند. ابتدا کلاس سرور و سپس کلاس کلاینت را بررسی خواهیم کرد.

Server.java

```
import java.io.*;
import java.net.*;

public class Server {

    public static void main(String[] args){
        try{
            ServerSocket ss=new ServerSocket(6666);
            Socket s=ss.accept();//ارتباط برقرار می کند
            DataInputStream dis=new DataInputStream(s.getInputStream());
            String str=(String)dis.readUTF();
            System.out.println("message= "+str);
            ss.close();
        }catch(Exception e){System.out.println(e);}
    }
}
```

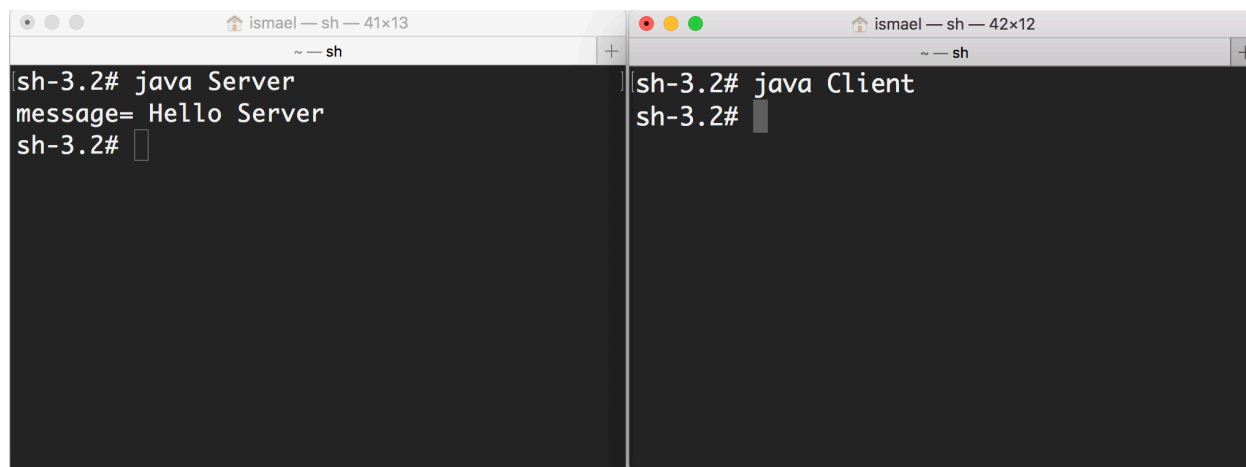
Client.java

```
import java.io.*;
import java.net.*;

public class Client {

    public static void main(String[] args) {
        try{
            Socket s=new Socket("localhost",6666);
            DataOutputStream dout=new DataOutputStream(s.getOutputStream());
            dout.writeUTF("Hello Server");
            dout.flush();
            dout.close();
            s.close();
        }catch(Exception e){System.out.println(e);}
    }
}
```

برای اجرای این برنامه دو خط فرمان باز کنید و هر برنامه را در هر خط فرمان همانطور که در شکل زیر نشان داده شده است اجرا کنید. پس از اجرای برنامه کلاینت، پیامی بر روی کنسول سرور نمایش داده می شود.



```
ismael — sh — 41x13
~ — sh
sh-3.2# java Server
message= Hello Server
sh-3.2#

ismael — sh — 42x12
~ — sh
sh-3.2# java Client
sh-3.2#
```

نمونه ای از برنامه نویسی سوکت جاوا (خواندن و نوشتن هر دو طرف)  
در این مثال، مشتری ابتدا روی سرور می نویسد سپس سرور متن را دریافت و چاپ می کند. سپس سرور برای مشتری می نویسد و مشتری متن را دریافت و چاپ می کند. مرحله ادامه دارد.

## Server.java

```
import java.net.*;
import java.io.*;

public class Server{

    public static void main(String args[]) throws Exception{
        ServerSocket ss=new ServerSocket(3333);
        Socket s=ss.accept();
        DataInputStream din=new DataInputStream(s.getInputStream());
        DataOutputStream dout=new DataOutputStream(s.getOutputStream());
        BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

        String str="",str2="";
        while(!str.equals("stop")){
            str=din.readUTF();
            System.out.println("client says: "+str);
            str2=br.readLine();
            dout.writeUTF(str2);
            dout.flush();
        }
        din.close();
        s.close();
        ss.close();
    }
}
```

## Client.java

```
import java.net.*;
import java.io.*;

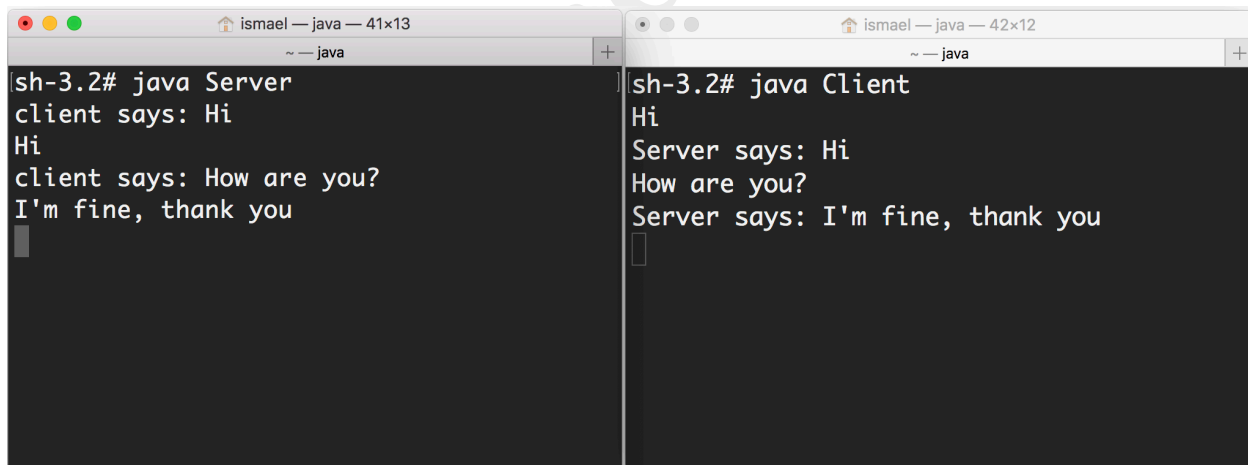
public class Client{

    public static void main(String args[]) throws Exception{
        Socket s=new Socket("localhost",3333);
        DataInputStream din=new DataInputStream(s.getInputStream());
        DataOutputStream dout=new DataOutputStream(s.getOutputStream());
        BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

        String str="",str2="";
        while(!str.equals("stop")){
            str=br.readLine();
            dout.writeUTF(str);
            dout.flush();
            str2=din.readUTF();
            System.out.println("Server says: "+str2);
        }

        dout.close();
        s.close();
    }
}
```

برای اجرای این برنامه دو خط فرمان باز کنید و هر برنامه را در هر خط فرمان همانطور که در شکل زیر نشان داده شده است اجرا کنید. پس از اجرای برنامه کلاینت و سرور پیغام ها را به صورت زیر وارد کنید.



```
sh-3.2# java Server
client says: Hi
Hi
client says: How are you?
I'm fine, thank you

sh-3.2# java Client
Hi
Server says: Hi
How are you?
Server says: I'm fine, thank you
```

## سوالات

۱. سوکت چیست و انواع آن را نام ببرید.
۲. آدرس آی پی و پورت را با ذکر مثال تعریف کنید.
۳. پروتکل چیست؟  
جواب: مراجعه به متن.
۴. جهت برقراری ارتباط بین کامپیوترها در یک شبکه، چه کارهایی را باید انجام داد.  
جواب:
  - ۱) آدرس ماشینی که می‌خواهیم اطلاعاتی از آن بگیریم یا به آن ارسال کنیم.
  - ۲) برنامه‌ای از آن ماشین که درخواست اطلاعات کرده؛ و یا اینکه می‌خواهیم اطلاعاتی از آن برنامه کسب کنیم.
۵. جهت برقراری ارتباط بین کامپیوترها در یک شبکه، چه کارهایی یک سوکت انجام می‌دهد.  
جواب:
  - ۱) اتصال به ماشین راه دور
  - ۲) ارسال داده‌ها
  - ۳) دریافت داده‌ها
  - ۴) بستن یا خاتمه‌ی اتصال
۶. تفاوت پورت و سوکت.  
جواب: اتصال یک سوکت همانطور که گفته شد، در حقیقت ترکیبی از آی پی یا نام هاست و یک شماره پورت از آن آی پی می‌باشد.
۷. خروجی دستور زیر چیست.

```
ServerSocket serverSocket = new ServerSocket(6666);  
Socket socket = serverSocket.accept();
```

جواب: برای ایجاد یک برنامه سرور سوکت در سرور لوکال "localhost" و پورت ۶۶۶۶ استفاده می‌شود.

۸. برنامه ای بنویسید که یک کلاینت بتواند با سرور چت کند (مثال عملی)  
جواب: مراجعه به متن.