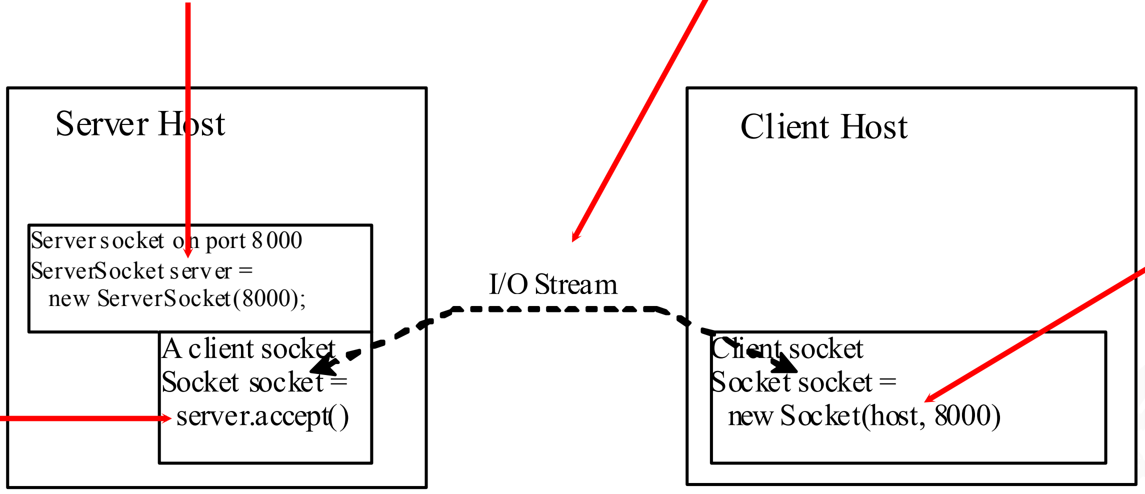
Networking Programming in Java

Client/Server Communications

The server must be running when a client starts. The server waits for a connection request from a client. To establish a server, you need to create a server socket and attach it to a port, which is where the server listens for connections.

After the server accepts the connection, communication between server and client is conducted the same as for I/O streams.



```
Server socket on port 8000
ServerSocket server =
  new ServerSocket(8000);
```

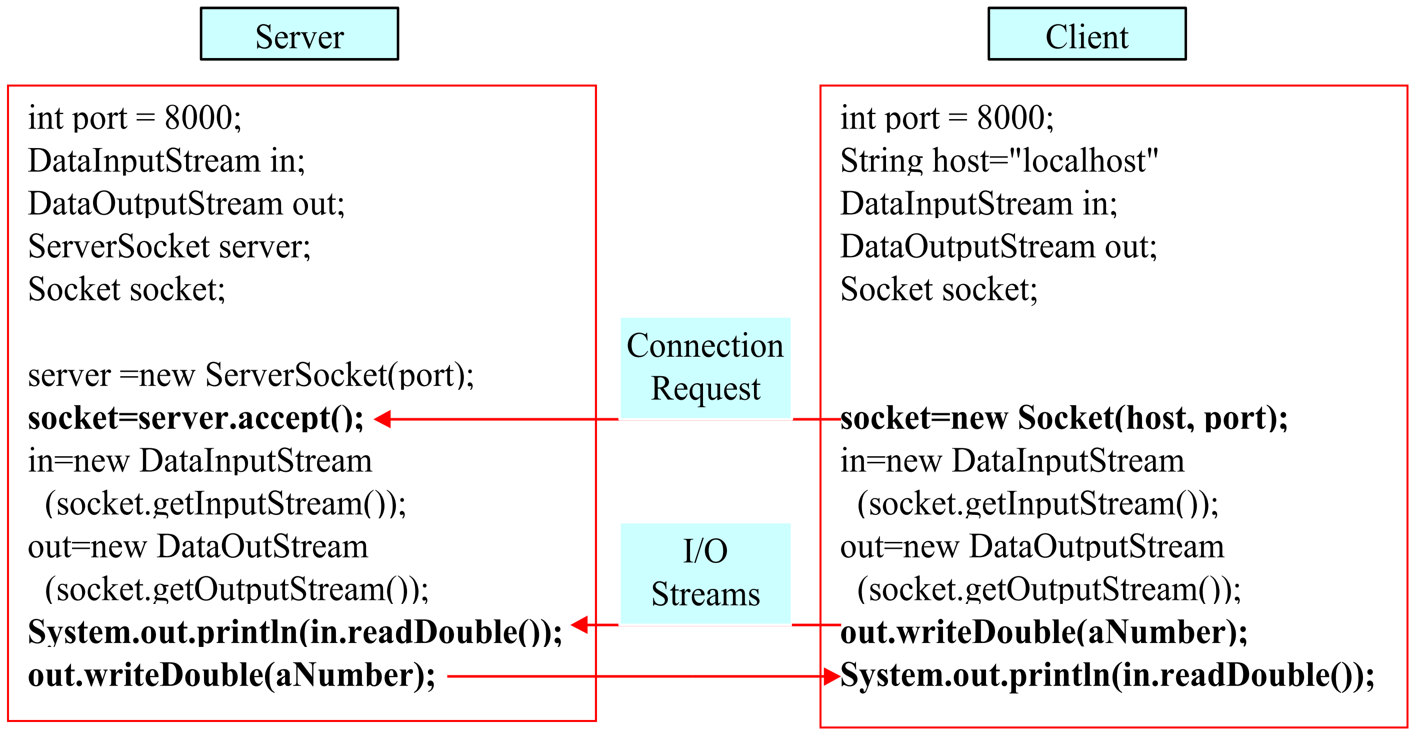
```
A client socket
Socket socket =
 server.accept()
```

```
Client socket
Socket socket =
  new Socket(host, 8000)
```

After a server socket is created, the server can use this statement to listen for connections.

The client issues this statement to request a connection to a server.

Data Transmission through Sockets



```
int port = 8000;
DataInputStream in;
DataOutputStream out;
ServerSocket server;
Socket socket;

server =new ServerSocket(port);
socket=server.accept();
in=new DataInputStream
  (socket.getInputStream());
out=new DataOutStream
  (socket.getOutputStream());
System.out.println(in.readDouble());
out.writeDouble(aNumber);
```

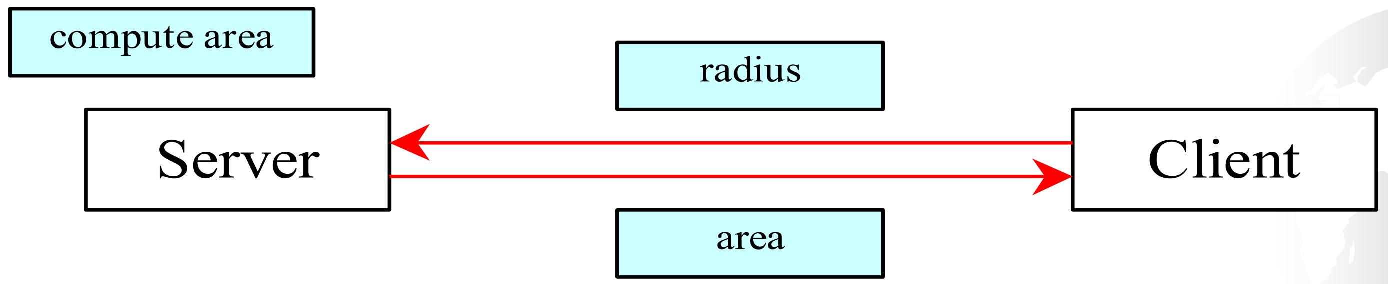
```
int port = 8000;
String host="localhost"
DataInputStream in;
DataOutputStream out;
Socket socket;


socket=new Socket(host, port);
in=new DataInputStream
  (socket.getInputStream());
out=new DataOutputStream
  (socket.getOutputStream());
out.writeDouble(aNumber);
System.out.println(in.readDouble());
```

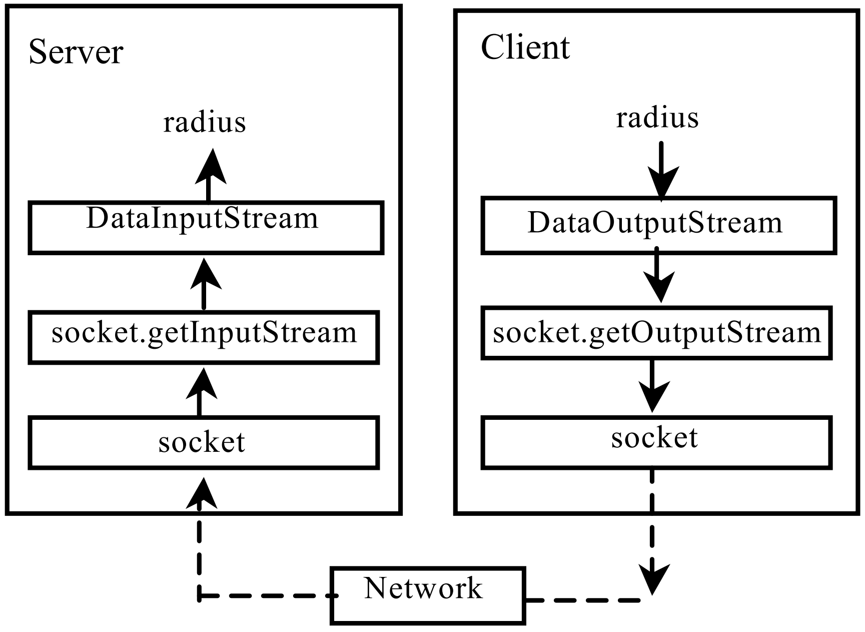
```
InputStream input = socket.getInputStream();
OutputStream output = socket.getOutputStream();
```

A Client/Server Example

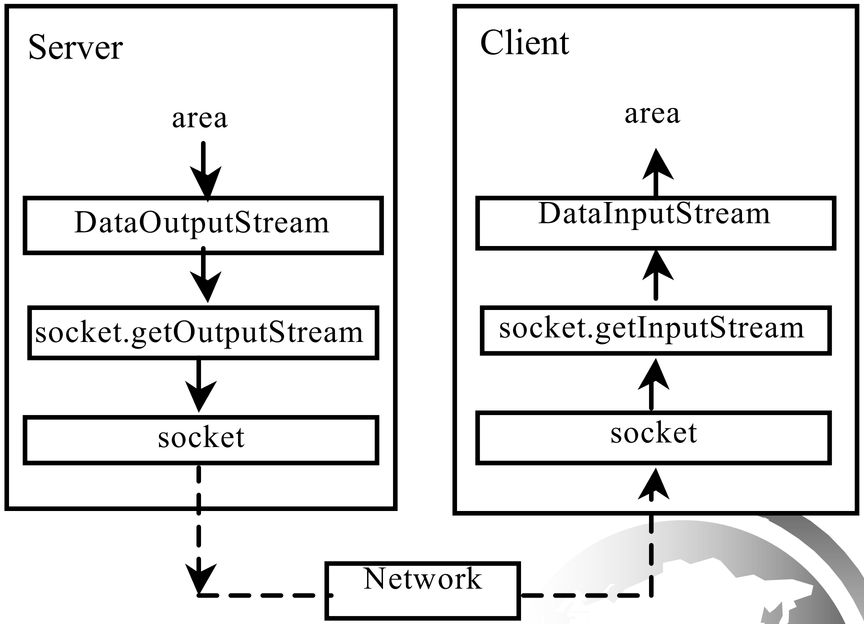
- Write a client to send data to a server.
- The server receives the data, uses it to produce a result, and then sends the result back to the client.
- The client displays the result on the console.
- In this example, the data sent from the client is the radius of a circle, and the result produced by the server is the area of the circle.



A Client/Server Example, cont.

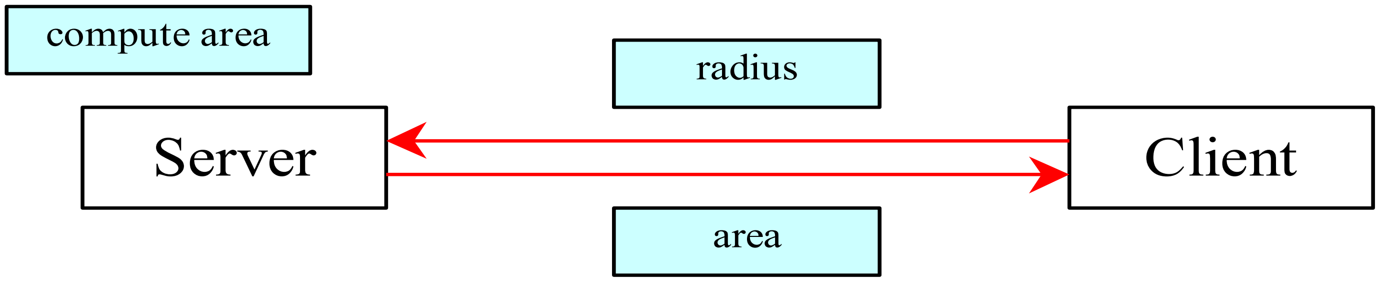


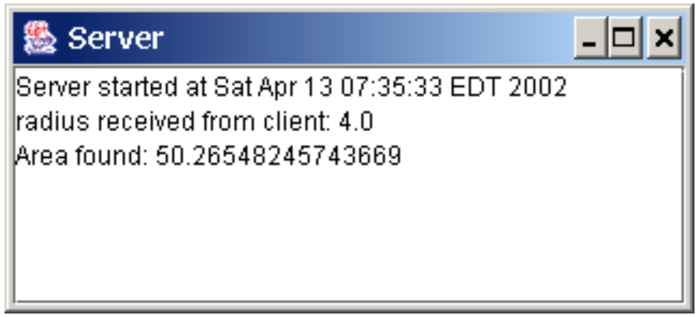
(A)



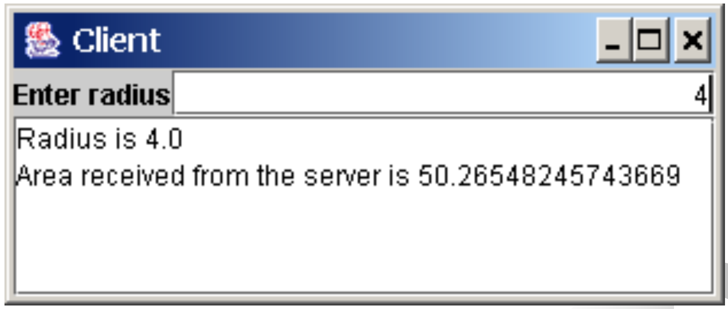
(B)

A Client/Server Example, cont. (*Server* and *Client*)





```
Server started at Sat Apr 13 07:35:33 EDT 2002
radius received from client: 4.0
Area found: 50.26548245743669
```



Enter radius: 4

```
Radius is 4.0
Area received from the server is 50.26548245743669
```

Note: Start the server, then the client.

The InetAddress Class

Occasionally, you would like to know who is connecting to the server. You can use the InetAddress class to find the client's host name and IP address.

The InetAddress class models an IP address. You can use the statement shown below to create an instance of InetAddress for the client on a socket.

```
InetAddress inetAddress = socket.getInetAddress();
```

Next, you can display the client's host name and IP address, as follows:

```
System.out.println("Client's host name is " +
                                inetAddress.getHostName());

System.out.println("Client's IP Address is " +
                                inetAddress.getHostAddress());
```

Serving Multiple Clients

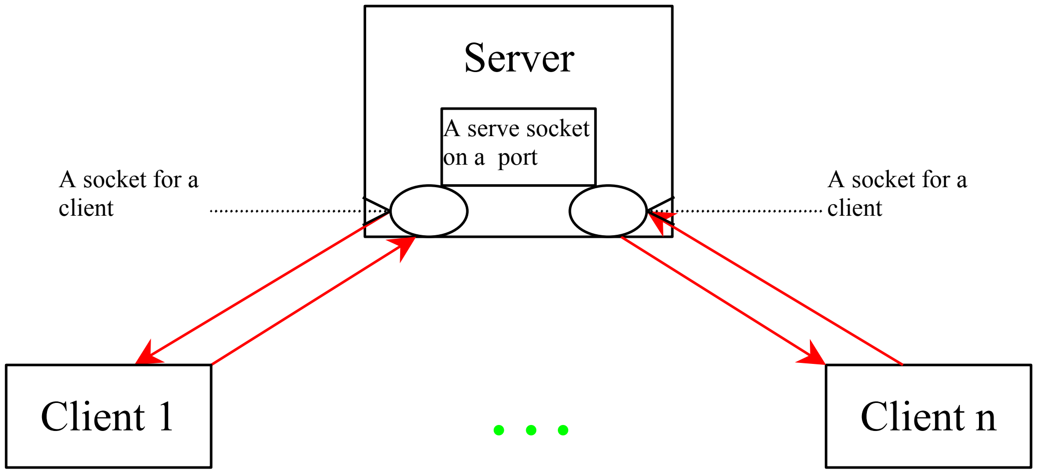
- ✦ Multiple clients are quite often connected to a single server at the same time.
- ✦ Typically, a server runs constantly on a server computer, and clients from all over the Internet may want to connect to it.
- ✦ You can use threads to handle the server's multiple clients simultaneously. Simply create a thread for each connection.

Here is how the server handles the establishment of a connection:

```
while (true) {
  Socket socket = serverSocket.accept();
  HandleTask task = new HandleTask(socket);
  Thread thread = new Thread(task);
  thread.start();
}
```

- ✦ The server socket can have many connections.
- ✦ Each iteration of the <u>while</u> loop creates a new connection.
- ✦ Whenever a connection is established, a new thread is created to handle communication between the server and the new client; and this allows multiple connections to run at the same time.

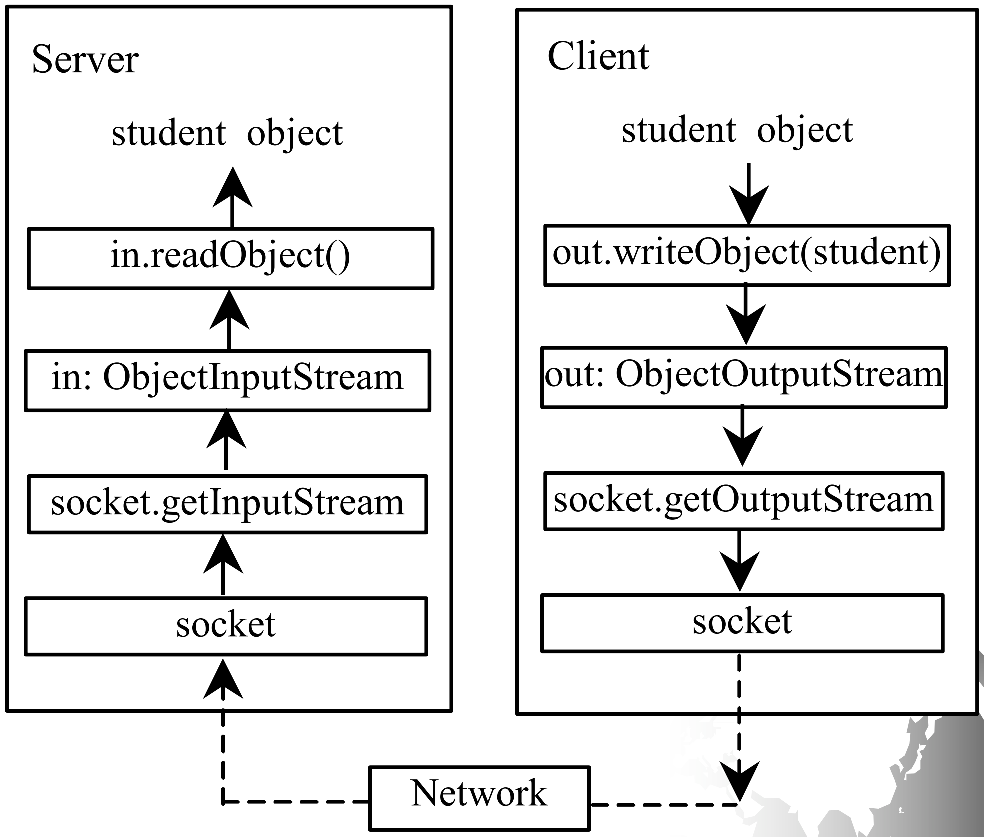
Serving Multiple Clients (Example – *MultiThreadServer, Client*)



Note: Start the server first, then start multiple clients.

Passing Objects in Network Programs (Example – *Student*, *StudentServer*, *StudentClient*)

- Write a program that collects student information from a client and send them to a server.
- Passing student information in an object.



Note: Start the server first, then the client.