ServerSocket ss = new ServerSocket(port);

ServerSocket Class

The ServerSocket class lets client programs connect with a server program. When a client connects, the server socket creates a Socket object, which the server can then use to communicate with the client.

ServerSocket() Creates a server socket that isn’t bound to any port.

ServerSocket(int port)

Creates a server socket and binds it to the specified port. Then the server socket listens for connection attempts on this port.

Socket clientSocket = ss.accept();

A socket is simply an endpoint for communications between the machines. The Socket class can be used to create a socket.

Socket accept()

Listens for connection attempts via the port this socket is bound to. The thread that calls this method waits until a connection is made. Then this method exits, returning a Socket object that can be used to communicate with the client.

\*\*\*\*\*\*\*\*\*\*\*\*\*88

System.out.println("Recieved connection from "+clientSocket.getInetAddress()+" on port "+clientSocket.getPort());

socket.getInetAddress() returns an InetAddress object that contains the IP address of the remote machine.

Port number of remote machine

//create two threads to send and recieve from client

RecieveFromClientThread recieve = new RecieveFromClientThread(clientSocket);

Thread thread = new Thread(recieve);

thread.start();

Creating a thread in Java is done like this:

Thread thread = new Thread();

To start the Java thread you will call its start() method, like this:

thread.start();

SendToClientThread send = new SendToClientThread(clientSocket);

Thread thread2 = new Thread(send);

thread2.start();

class RecieveFromClientThread implements Runnable

**Runnable:** After a newly born thread is started, the thread becomes runnable. A thread in this state is considered to be executing its task.

Socket clientSocket=null;

BufferedReader recieve=null;

BufferedReader

Reads text from a character-input stream, buffering characters so as to provide for the efficient reading of characters, arrays, and lines

public RecieveThread(Socket sock) {

this.sock = sock;

}//end constructor

As a first step you need to implement a run() method provided by **Runnable**interface. This method provides entry point for the thread and you will put you complete business logic inside this method. Following is simple syntax of run() method:

public void run() {

try{

recieve = new BufferedReader(new InputStreamReader(this.sock.getInputStream()));//get inputstream

String msgRecieved = null;

while((msgRecieved = recieve.readLine())!= null)

{

System.out.println("From Server: " + msgRecieved);

System.out.println("Please enter something to send to server..");

}

}catch(Exception e){System.out.println(e.getMessage());}

}//end run

.