```
import java.net.* ;
import java.util.* ;

public class EchoClient {

	static final int ECHOPORT = 8000;
	static final String LOCALSERVER = "localhost";

	public static void main(String[] args) throws Exception {
		if( args.length > 1 ) {
			System.err.printf("usage: java EchoClient [server_computer] \n") ;
			System.exit(0);
		}
		// Get server IP address and create an UDP socket
		String servidor = LOCALSERVER;
		if( args.length == 1) servidor = args[0];
		InetAddress serverAddress = InetAddress.getByName( servidor ) ;
		DatagramSocket socket = new DatagramSocket();

		// Read a message from the input console
		Scanner in = new Scanner( System.in );
		System.out.printf("Echo client - please type a messsage: ") ;
		String request = in.nextLine();

		// Prepare a datagram to send to the server in a compact way,
		// send it, wait for the reply, print it
		DatagramPacket datagram = new DatagramPacket ( request.getBytes(),
			request.length(), serverAddress, ECHOPORT ) ;
		socket.send( datagram );
		socket.receive( datagram );
		socket.close() ;
		System.out.printf("Got: \"%s\"\n",
			new String( datagram.getData(), 0, datagram.getLength() ) ) ;
	} // main
}
```

```
import java.net.*;

public class EchoServer  {

	static final int ECHOPORT = 8000;
	static final int MAXBLOCKSIZE = 1460 ;

	public static void main(String[] args) throws Exception {

		// create input / output UDP socket and bind it to the ECHOPORT
		DatagramSocket socket = new DatagramSocket( ECHOPORT ) ;

		// prepare a datagram, for receiving and sending
		DatagramPacket datagram = new DatagramPacket(
			new byte[MAXBLOCKSIZE], MAXBLOCKSIZE ) ;

		for(;;) { // server endless loop
			System.out.println( "Echo server ready");
			socket.receive( datagram );
			System.out.printf("Got: %s \n", new String( datagram.getData(), 0,
				datagram.getLength()));
			//prepare an UDP datagram with the reply
			// the port and IP address associated with the datagram are the ones
			// of the sender when a datagram is received and are used as the
			// IP address and port of the destination when the datagram is sent
			socket.send( datagram ) ;
			datagram.setLength(MAXBLOCKSIZE); // since the payload
								// may have shrunk
		} // endless loop
	} // main
} // EchoServer
```