java.net / Sockets TCP / Servidor

```
package aula0;

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

public class TcpServer {
    public static void main(String[] args) throws Exception {
        ServerSocket ss = new ServerSocket( 9000 ) ;
        while( true ) {
            Socket cs = ss.accept() ;
            System.out.println("Connection from: " + cs.getInetAddress()+ ":" + cs.getPort() ) ;

            InputStream is = cs.getInputStream() ;
            byte[] buffer = new byte[1024] ;
            int n ;
            while( (n = is.read( buffer )) != -1 ) {
                System.out.write( buffer, 0, n ) ;
            }
            cs.close() ;
            System.out.println("Connection closed.") ;
        }
    }
}
```

java.net / Sockets TCP / Cliente

```
package aula0;

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

public class TcpClient {
    public static void main(String[] args) throws Exception {
        String input ;

        Socket cs = new Socket( args[0], 9000 ) ;
        OutputStream os = cs.getOutputStream() ;
        do {
            input = readLine() + "\n";
            os.write( input.getBytes() ) ;
        } while( ! input.equals(".\n") ) ;
        os.close() ;
        cs.close() ;
    }

    public static String readLine() throws Exception {
        return new BufferedReader( new InputStreamReader( System.in ) ).readLine() ;
    }
}
```

java.net / Sockets UDP / Servidor

```
package aula0;

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

public class UdpServer {

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

        DatagramSocket socket = new DatagramSocket( 9000 ) ;
        while( true ) {
            byte[] buffer = new byte[65536] ;
            DatagramPacket packet = new DatagramPacket( buffer, buffer.length ) ;
            socket.receive( packet ) ;
            System.out.println("Message from:"+ packet.getAddress() + ":"+ packet.getPort());
            System.out.write( packet.getData(), 0, packet.getLength() ) ;
        }
    }
}
```

java.net / Sockets UDP / Cliente

```
package aula0;

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

public class UdpClient {
    public static void main(String[] args) throws Exception {
        final int port = 9000 ;
        final InetAddress address = InetAddress.getByName( args[0] ) ;

        DatagramSocket socket = new DatagramSocket() ;
        do {
            byte[] input = (readLine() + "\n").getBytes();
            DatagramPacket packet = new DatagramPacket( input, input. length ) ;
            packet.setAddress( address ) ;
            packet.setPort( port ) ;
            socket.send( packet ) ;
        } while( ! input.equals(".\n") ) ;
        socket.close() ;
    }

    public static String readLine() throws Exception {
        return new BufferedReader( new InputStreamReader( System.in ) ).readLine() ;
    }
}
```

java.net / Sockets Multicast / Servidor

```
package aula0;

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

public class MulticastServer {
    public static void main(String[] args) throws Exception {
        final InetAddress address = InetAddress.getByName( args[0] ) ;
        if( ! address.isMulticastAddress()) {
          System.out.println( "Use range : 224.0.0.0 -- 239.255.255.255");
          System.exit( 1);
        }
        MulticastSocket socket = new MulticastSocket( 9000 ) ;
        socket.joinGroup( address);
        while( true ) {
            byte[] buffer = new byte[65536] ;
            DatagramPacket packet = new DatagramPacket( buffer, buffer.length ) ;
            socket.receive( packet ) ;
            System.out.write( packet.getData(), 0, packet.getLength() ) ;
        }
    }
}
```

java.net / Sockets Multicast / Cliente

```
package aula0;

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

public class MulticastClient {
    public static void main(String[] args) throws Exception {
        final int port = 9000 ;
        final InetAddress address = InetAddress.getByName( args[0] ) ;
        if( ! address.isMulticastAddress()) {
          System.out.println( "Use range : 224.0.0.0 -- 239.255.255.255");
        }
        MulticastSocket socket = new MulticastSocket() ;
        do {
            byte[] input = (readLine() + "\n").getBytes();
            DatagramPacket packet = new DatagramPacket( input, input.length ) ;
            packet.setAddress( address ) ;
            packet.setPort( port ) ;
            socket.send( packet ) ;
        } while( ! input.equals(".\n") ) ;
        socket.close() ;
    }
    public static String readLine() throws Exception {
        return new BufferedReader( new InputStreamReader( System.in ) ).readLine() ;
    }
```

java.net / Servidor TCP com múltiplos threads / Servidor

```
package aula0;

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

public class MultiThreadedTcpServer {
    public static void main(String[] args) throws Exception {
        ServerSocket ss = new ServerSocket( 9000 ) ;
        while( true ) {
            new Thread( new ConnectionHandler( ss.accept() ) ).start() ;
        }
    }
}

class ConnectionHandler implements Runnable {
    private Socket cs ;

    ConnectionHandler( Socket cs ) {
        this.cs = cs ;
    }

    public void run() {
        try {
            final int srcPort = cs.getPort() ;
            final InetAddress srcAddress = cs.getInetAddress() ;
            System.out.println("Accepted connection from : " +  srcAddress + ":" + srcPort ) ;

            int n ;
            byte[] buffer = new byte[1024] ;
            while( (n = cs.getInputStream().read( buffer ) ) != -1 ) {
                System.out.write( buffer, 0, n ) ;
            }
            cs.close() ;
        } catch( Exception x ) {
            x.printStackTrace() ;
        }
    }
}
```