

Módulo 13

Socket

Programação Orientada a Objetos

Java

(Rone Ilídio)

O que é socket?

- **Definição:**

Interface da camada de transporte que permite comunicação entre dois processos que podem estar em computadores distintos.

- **Utilidade:**

Comunicação entre máquinas conectadas em uma rede.

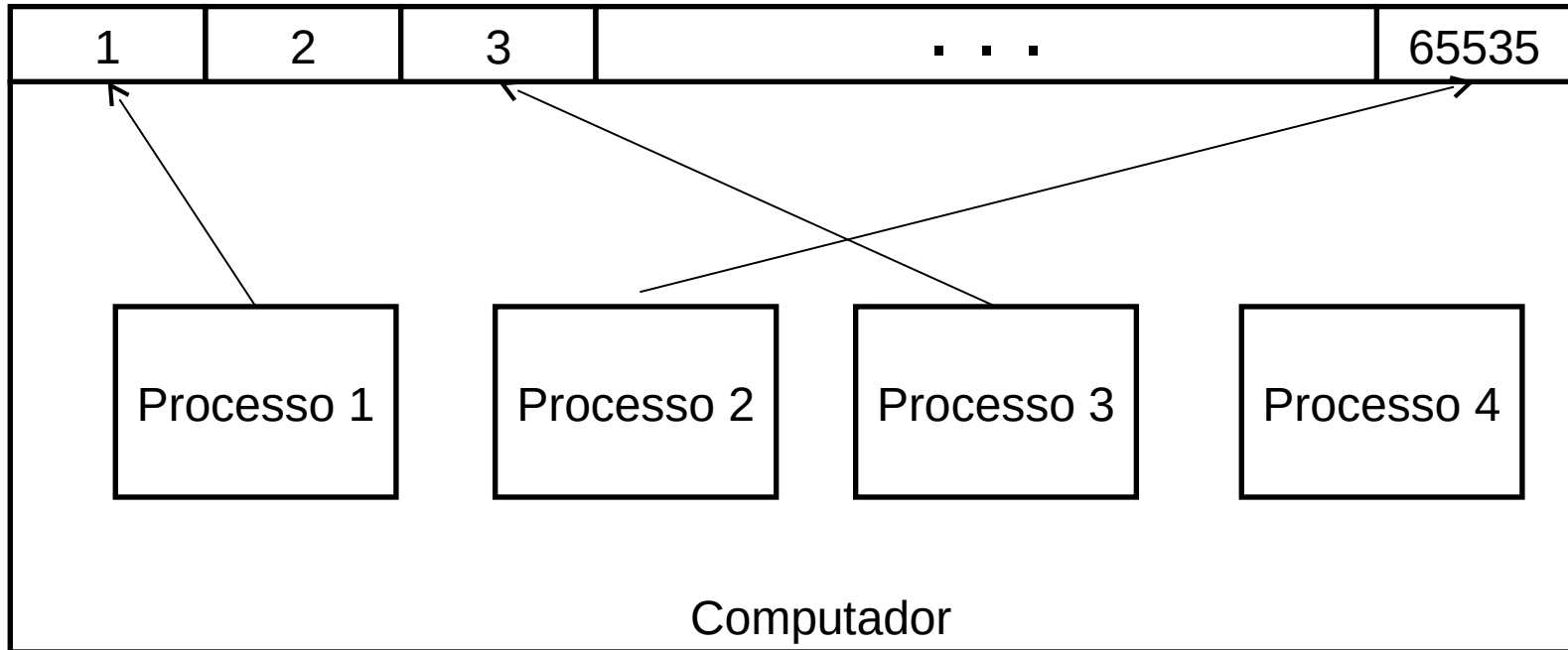
Endereço IP

- Número que identifica todos os computadores conectados à internet
- Exemplo: 255.255.255.255 (obs: ipv4)
- Como saber o ip no Windows → no prompt de comandos digite “ipconfig”



Portas

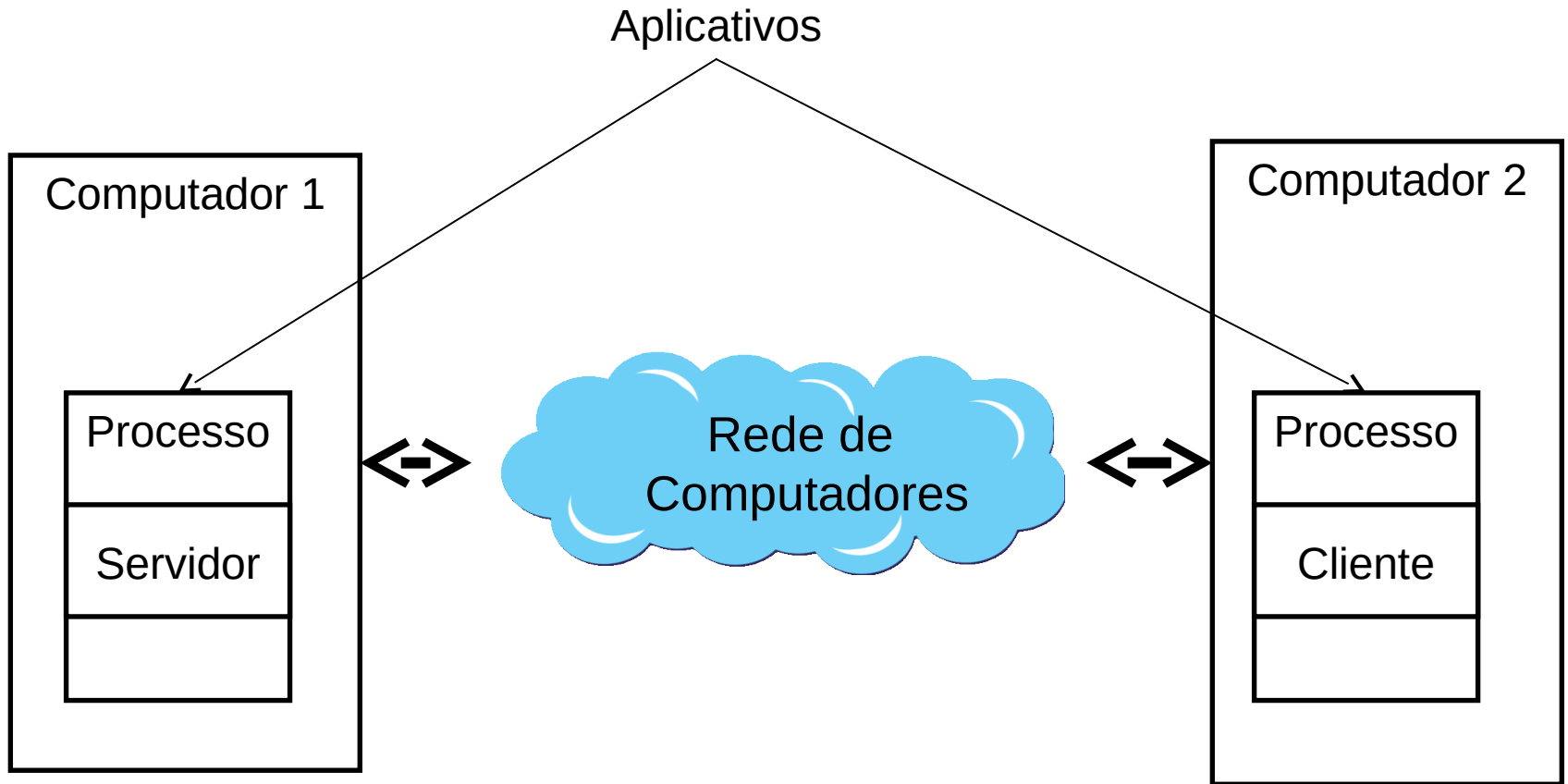
- Número de 16 bits que identifica o processo
- Canais por onde as aplicações se comunicam
- Portas de 0 a 1023: reservadas



Class Importantes

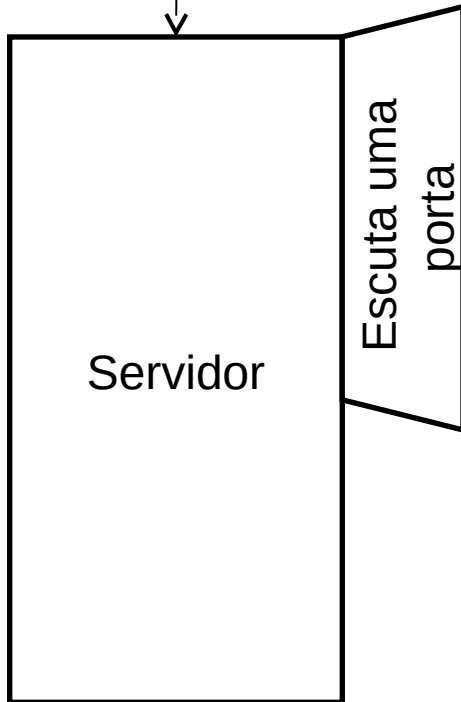
- Pacote java.net
- **ServerSocket**: define objetos servidores, os quais abrirão um canal de comunicação e esperarão conexões externas
- **Socket**: define objetos que formarão o link (um no cliente outro no servidor)
- Obs: a conexão não depende da linguagem de programação

Exemplo de conexão

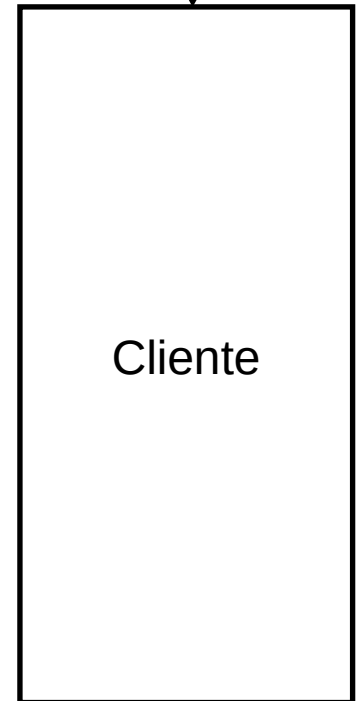


Exemplo de conexão

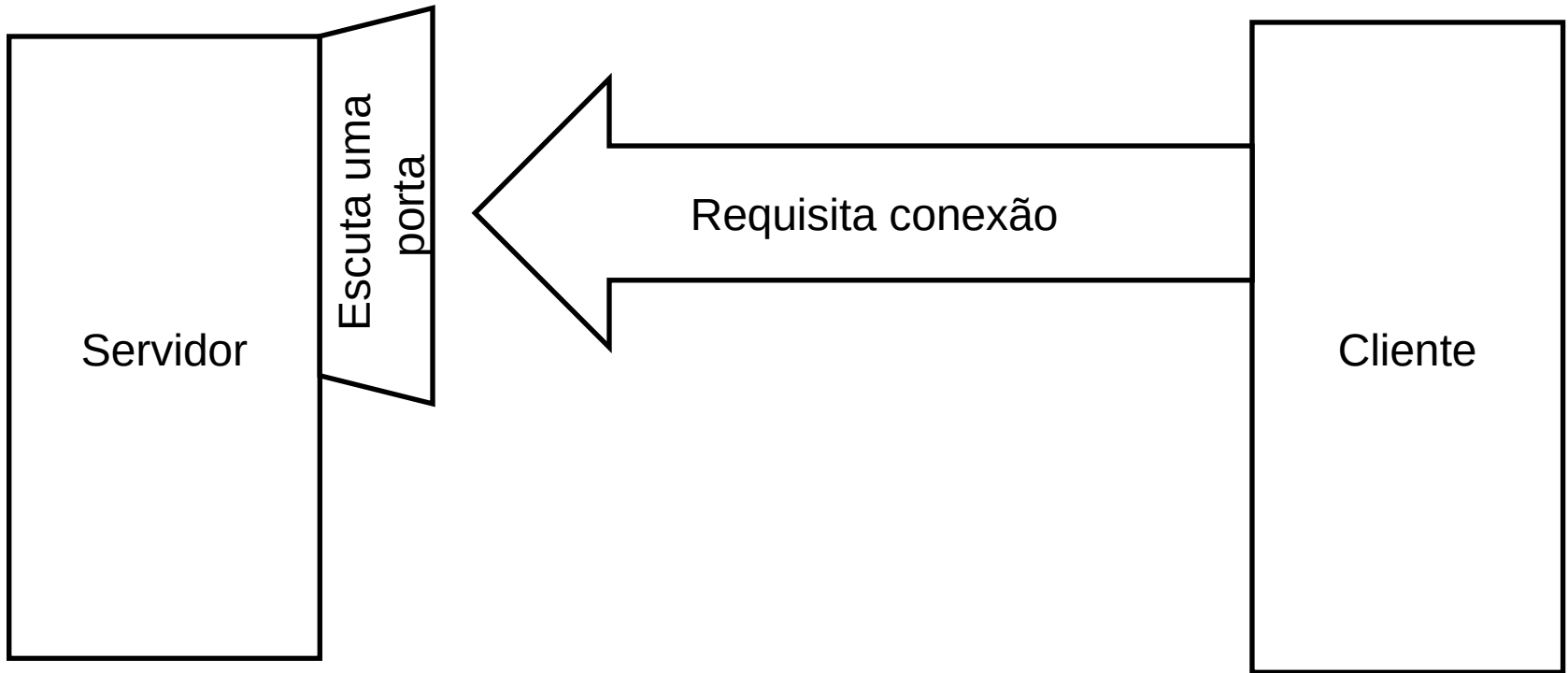
Objeto do tipo
ServerSocket



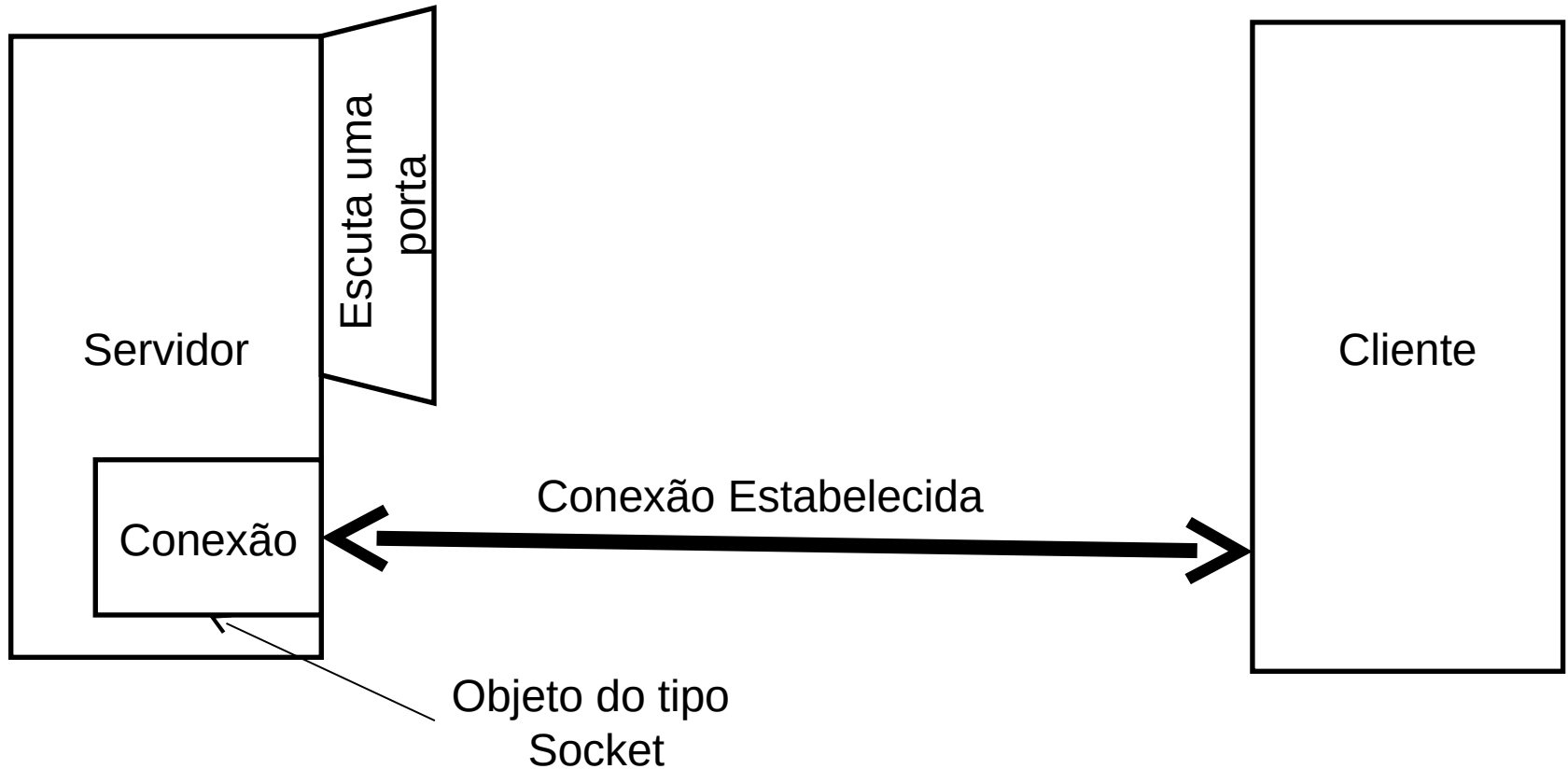
Objeto do tipo
Socket



Exemplo de conexão



Exemplo de conexão



Exemplo de UDPSocket

- Duas classes
 - UDPServer: cria um servidor UDP que recebe as mensagens e exibe na tela
 - UDPClient: envia para o servidor mensagens fornecidas pelo usuário
- Se o usuário digitar fim, tanto o servidor quanto o cliente são finalizados

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

class UDPServer {

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

        System.out.println("Servidor Iniciado...");

        while (true) {

            byte[] receiveData = new byte[1024];

            byte[] sendData = new byte[1024];

            DatagramSocket serverSocket = new DatagramSocket(9876);

            DatagramPacket receivePacket =

                new DatagramPacket(receiveData, receiveData.length);

            serverSocket.receive(receivePacket);

            String sentence = new String(receivePacket.getData());

            InetAddress IPAddress = receivePacket.getAddress();

            System.out.println("Recebido de " +

                IPAddress.getHostAddress()+":" + sentence);

            serverSocket.close();

            if (sentence.substring(0,3).equals("fim")) break;

        }

    }

}
```

```
package basic0,
import java.io.*;
import java.net.*;
import javax.swing.JOptionPane;
class UDPClient {
    public static void main(String args[]) throws Exception {
        while (true) {
            DatagramSocket clientSocket = new DatagramSocket();
            byte[] sendData = new byte[1024];
            byte[] receiveData = new byte[1024];
            InetAddress IPAddress = InetAddress.getByName("localhost");
            String sentence = JOptionPane.showInputDialog("Digite:");
            sendData = sentence.getBytes();
            DatagramPacket sendPacket =
                new DatagramPacket(sendData, sentence.length(), IPAddress, 9876);
            clientSocket.send(sendPacket);
            clientSocket.close();
            if (sentence.equals("fim")) break;
        }
    }
}
```

Exemplo de TCPSocket

- Sequência para criar um servidor
 - Criar um objeto do tipo ServerSocket
 - Fazer esse objeto escutar uma porta
 - Ao escutar uma conexão, criar um objeto do tipo Socket
 - Criar objetos de escrita e leitura e associar ao objeto Socket
 - O servidor reenvia todas as mensagens recebidas.
- O cliente
 - Cria a conexão
 - Envia uma mensagem digitado pelo usuário
 - Recebe e exibe a resposta
- Quando o usuário digitar “fim”, ambos terminam

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

public class TCPServer {

public static void main(String args[]) {

    System.out.println("Servidor iniciado ...");

    ServerSocket socketServidor = null;

    try {

        socketServidor = new ServerSocket(5000);

        Socket socketCliente = null;

        socketCliente = socketServidor.accept();

        PrintWriter out =

            new PrintWriter(socketCliente.getOutputStream(), true);

        BufferedReader in =

            new BufferedReader(new InputStreamReader(socketCliente.getInputStream()));

        String entrada;

        while ((entrada = in.readLine()) != null) {

            System.out.println("Recebido:" + entrada);

            out.println(entrada);

            if (entrada.equals("fim")) break;

        }

        out.close();

        in.close();

        socketServidor.close();

        socketCliente.close();

    } catch (IOException e) {

        System.out.println("Erro ao criar os streams");

        System.exit(1);

    }

}

}
```

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

public class TCPClient {

    public static void main(String args[]) {

        Socket socket = null; // Socket cliente
        PrintWriter out = null; // Escreve no socket
        BufferedReader in = null; // Le do socket
        try {
            socket = new Socket("localhost", 5000);
            out = new PrintWriter(socket.getOutputStream(), true);
            in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
            String entradaservidor = "";
            String entradausuario =
                JOptionPane.showInputDialog("Digite para enviar para o servidor:");
            while (!entradausuario.equals("fim")) {
                out.println(entradausuario);
                entradaservidor = in.readLine();
                entradausuario = JOptionPane.showInputDialog(
                    "Chegou do servidor: " + entradaservidor + "\nDigite para enviar para o servidor:");
            }
            out.close();
            in.close();
            socket.close();
        } catch (IOException e) {
            System.err.println("Erro na criação dos objetos");
            System.exit(1);
        }
    }
}
```