UNIVERSIDADE FEDERAL RURAL DO SEMIÁRIDO - UFERSA
Departamento de Computação - DC
Graduação em Ciência da Computação
Disciplina: Sistemas Distribuídos
Prof.: Paulo Henrique Lopes Silva

Prática 6

1. Servidor de Eco com threads.

- Classe Cliente.java

```
import java.io.IOException;
import java.io.PrintStream;
import java.net.Socket;
import java.util.Scanner;

public class Cliente implements Runnable{

	private Socket cliente;
	private boolean conexao = true;

	public Cliente(Socket c){
		this.cliente = c;
	}

	public void run() {

		try {
			PrintStream saida;
			System.out.println("O cliente conectou ao servidor");

			//Prepara para leitura do teclado

			Scanner teclado = new Scanner(System.in);

			//Cria  objeto para enviar a mensagem ao servidor

			saida = new PrintStream(this.cliente.getOutputStream());

			//Envia mensagem ao servidor
			String snd;
			while(conexao){

				System.out.println("Digite uma mensagem: ");
				snd = teclado.nextLine();

				if (snd.equalsIgnoreCase("fim"))
					conexao = false;
				else
					System.out.println(snd);

				saida.println(snd);

			}
```

```
                        saida.close();
                        teclado.close();
                        cliente.close();
                        System.out.println("Cliente finaliza conexão.");

                } catch (IOException e) {
                        e.printStackTrace();
                }
        }

}
```

- Classe RodaCliente.java

```
import java.io.IOException;
import java.net.InetAddress;
import java.net.Socket;
import java.net.UnknownHostException;

public class RodaCliente {

        public static void main(String args[]) throws UnknownHostException, IOException {

                /** para se conectar ao servidor, cria-se objeto Socket.
                O primeiro parâmetro é o IP ou endereço da máquina que
                se quer conectar e o segundo é a porta da aplicação.
                Neste caso, usa-se o IP da máquina local (127.0.0.1)
                e a porta da aplicação ServidorDeEco (12345). */

                Socket socket = new Socket("127.0.0.1", 12345);
                InetAddress inet = socket.getInetAddress();
                System.out.println("HostAddress = "+inet.getHostAddress());
                System.out.println("HostName = "+inet.getHostName());

                /**Cria um novo objeto Cliente com a conexão socket para que seja executado em um novo
processo. Permitindo assim a conexão de vário clientes com o servidor.*/

                Cliente c = new Cliente(socket);
                Thread t = new Thread(c);
                t.start();
        }

}
```

- Classe Servidor.java

```
package servidor;

import java.io.IOException;
import java.net.Socket;
import java.util.Scanner;

public class Servidor implements Runnable{

        public Socket socketCliente;
```

```
	public static int cont = 0;

	public Servidor(Socket cliente){
		this.socketCliente = cliente;
	}

	/* A classe Thread, que foi instancia no servidor, implementa Runnable.
	   Então você terá que implementar sua lógica de troca de mensagens dentro deste método 'run'. */

	 public void run(){
		System.out.println("Conexao "+Servidor.cont+" com o cliente " +
this.socketCliente.getInetAddress().getHostAddress() +"/" +this.socketCliente.getInetAddress().getHostName());

		try {
			Scanner s = null;
			s = new Scanner(this.socketCliente.getInputStream());
			String rcv;

			//Exibe mensagem no console
			while(s.hasNextLine()){
				rcv = s.nextLine();

				if (rcv.equalsIgnoreCase("fim"))
					break;
				else
					System.out.println(rcv);
			}

			//Finaliza scanner e socket
			s.close();
			System.out.println("Fim do cliente "+this.socketCliente.getInetAddress());
			this.socketCliente.close();


		} catch (IOException e) {
			e.printStackTrace();
		}
	}

}
```

- Classe RodaServidor.java

```
package servidor;

import java.net.InetAddress;
import java.net.ServerSocket;
import java.net.Socket;

public class RodaServidor {

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

		//Cria um socket na porta 12345
		ServerSocket socketServidor = new ServerSocket(12345);
```

```
                System.out.println("Servidor rodando na porta "+socketServidor.getLocalPort());
                System.out.println("HostAddress = "+ InetAddress.getLocalHost().getHostAddress());
                System.out.println("HostName = "+ InetAddress.getLocalHost().getHostName());

                /* Aguarda alguém se conectar. A execução do servidor
                 fica bloqueada na chamada do método accept da classe
                ServerSocket. Quando alguém se conectar ao servidor, o
                método desbloqueia e retorna com um objeto da classe
                Socket, que é uma porta da comunicação. */

                System.out.println("Aguardando conexão do cliente...");

                while (true) {

                        Socket cliente = socketServidor.accept();
                        // Cria uma thread do servidor para tratar a conexão
                        Servidor servidor = new Servidor(cliente);
                        Thread t = new Thread(servidor);
                        // Inicia a thread para o cliente conectado
                        Servidor.cont++;
                        t.start();
                }

        }

}
```

Exercícios:

1. Crie um projeto no Eclipse, codifique, execute e observe o funcionamento dessa aplicação.

2. Transforme o servidor de Eco em um transmissor de mensagens de forma que quando dois clientes se conectem ao servidor, eles possam trocar mensagens entre si.