Instituto de Ciências Matemáticas e de Computação - ICMC/USP

Redes de Computadores
Atividade II
Prof. Jó Ueyama

Nome: Pedro Fernando Christofoletti dos Santos

No. USP: 11218560

1)

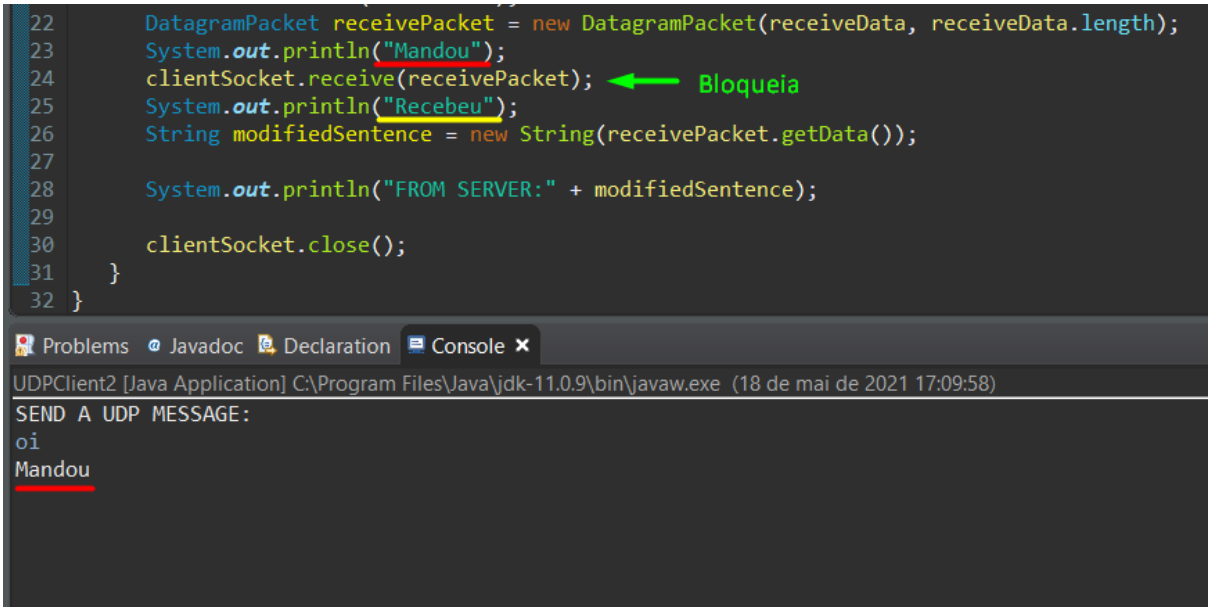
a) Suponha que você rode o TCPClient antes de rodar o TCPServer. O que acontece? Por que?

R: Uma exceção é lançada devido a um erro de conexão recusada. Isso acontece porque o protocolo TCP é orientado a conexão e se o cliente rodar antes do servidor então não será possível estabelecer uma conexão já que o servidor ainda não está escutando por pacotes.

```
Exception in thread "main" java.net.ConnectException: Connection refused: connect
        at java.base/java.net.PlainSocketImpl.connect0(Native Method)
        at java.base/java.net.PlainSocketImpl.socketConnect(PlainSocketImpl.java:101)
        at java.base/java.net.AbstractPlainSocketImpl.doConnect(AbstractPlainSocketImpl.java:399)
        at java.base/java.net.AbstractPlainSocketImpl.connectToAddress(AbstractPlainSocketImpl.java:242)
        at java.base/java.net.AbstractPlainSocketImpl.connect(AbstractPlainSocketImpl.java:224)
        at java.base/java.net.SocksSocketImpl.connect(SocksSocketImpl.java:403)
        at java.base/java.net.Socket.connect(Socket.java:608)
        at java.base/java.net.Socket.connect(Socket.java:557)
        at java.base/java.net.Socket.<init>(Socket.java:453)
        at java.base/java.net.Socket.<init>(Socket.java:230)
        at TCP.TCPClient.main(TCPClient.java:13)
```

b) Suponha que você rode o UDPClient antes de rodar o UDPServer. O que acontece? Por que?

R: Nenhuma exceção é lançada pois como o protocolo UDP não é orientado a conexão ele não requer que o servidor esteja ativo para enviar pacotes e também não garante que os pacotes cheguem. Entretanto, da maneira que foi implementado o programa ficará travado na função “receive” pois ela bloqueia a execução até que algum pacote seja retornado



c) O que acontece se você utilizar números de portas diferentes para o lado do cliente e do servidor?

R: Acontece o mesmo efeito que acontece com o servidor sendo executado após o cliente, já que o cliente está mandando os pacotes para um processo diferente, ou seja, para uma porta diferente, não haverá comunicação com o servidor ativo, logo, o TCPClient irá disparar uma exceção de conexão recusada e o UDPClient irá travar na função "receive" pois não receberá nenhuma pacote de retorno.

2) Suponha que em UDPClient.java, a linha

DatagramSocket clientSocket = new DatagramSocket(); Seja substituída por

DatagramSocket clientSocket = new DatagramSocket(5432);

a) Será necessário mudar UDPServer.java? Por quê?

R: Não será necessário mudar a classe UDPServer pois não importa em qual porta seja criado o socket do cliente contanto que o pacote seja criado usando a porta correta.

3) Agora implemente um servidor que recebe dois valores *a* e *b* de um cliente usando o mesmo esqueleto do código provido TCPClient e TCPServer. O servidor efetua a soma de *a* e *b* e devolve a soma ao servidor. Ambos o cliente e o servidor usam

sockets TCP para se comunicarem. O número de porta fica a critério do desenvolvedor e pode ser um número aleatório como 8000.

R:

MAIN

```
package application;

import TCP.TCPClient;
import TCP.TCPServer;

public class Main {

	public static void main(String[] args) throws Exception {
		TCPServer tcpServer = new TCPServer();
		tcpServer.start();
		long time = System.currentTimeMillis();
		//Delay de 1 segundo para iniciar servidor
		while(System.currentTimeMillis() < time+1000L) { }
		try {
			TCPClient tcpClient = new TCPClient();
			tcpClient.sendNumbers();
		}catch(Exception e) {
			System.out.println("Não foi possível realizar a
conexão :(");
		}
	}

}
```

TCPClient

```
package TCP;

import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.Socket;
import java.net.UnknownHostException;

public class TCPClient {

	private String number1, number2;
	private String sum;
	private BufferedReader inFromUser = new BufferedReader(new
InputStreamReader(System.in));
	private Socket clientSocket;
	private DataOutputStream outToServer;
	private BufferedReader inFromServer;

	public TCPClient() throws UnknownHostException, IOException {
		this.clientSocket = new Socket("localhost", 6789);
		this.outToServer = new
DataOutputStream(clientSocket.getOutputStream());
		this.inFromServer = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
	}

	public void sendNumbers() throws Exception {
		String command = "y";
		while (command.equals("y")) {
			System.out.print("Digite o primeiro número: ");
			number1 = inFromUser.readLine();
			System.out.print("Digite o segundo número: ");
			number2 = inFromUser.readLine();
			outToServer.writeBytes(number1 + ',' + number2 + '\n');
			sum = inFromServer.readLine();
			System.out.println("\nSoma do servidor: " + sum + "\n");
			System.out.print("Deseja efetuar outra soma? (y/n): ");
			command = inFromUser.readLine();
		}
		outToServer.writeBytes("end"+'\n');
		outToServer.close();
		inFromServer.close();
		inFromUser.close();
		clientSocket.close();
	}
}
```

TCPServer

```
package TCP;

import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.ServerSocket;
import java.net.Socket;

public class TCPServer extends Thread {

	private String clientSentence;
	private ServerSocket welcomeSocket;
	private Socket connectionSocket;
	private BufferedReader inFromClient;
	private DataOutputStream outToClient;
	private boolean running;

	public TCPServer() throws IOException {
		this.welcomeSocket = new ServerSocket(6789);
	}

	public void run() {
		try {
			connectionSocket = welcomeSocket.accept();
			inFromClient = new BufferedReader(new
InputStreamReader(connectionSocket.getInputStream()));
			outToClient = new DataOutputStream(connectionSocket.getOutputStream());
		} catch (IOException e) {
			e.printStackTrace();
		}
		running = true;
		while (running) {
			try {
				clientSentence = inFromClient.readLine();
				String[] numbers = clientSentence.split("[,]");
				if(numbers[0].contains("end")) {
					running = false;
				}else {
					try {
						int number1 = Integer.parseInt(numbers[0]);
						int number2 = Integer.parseInt(numbers[1]);
						int sum = number1 + number2;
						outToClient.writeBytes(String.valueOf(sum)+'\n');
					}catch(Exception e1) {
						outToClient.writeBytes("Valor inválido, digite apenas
números inteiros"+'\n');
					}
				}
			} catch (IOException e2) {
				e2.printStackTrace();
			}

		}
		try {
			inFromClient.close();
			outToClient.close();
			connectionSocket.close();
			welcomeSocket.close();
		} catch (IOException e3) {
			e3.printStackTrace();
		}

	}
}
```