Politécnico de Coimbra

Escola Superior de Tecnologia e Gestão

Licenciatura em Engenharia Informática

Sistemas Distribuídos

Ano Letivo 2021/22

Ficha 4

Elaborado em: 2022/03/29

Daniel Santos | Nº2019133865

Índice

List of Figures

1. Exercício 1

Execução do programa:

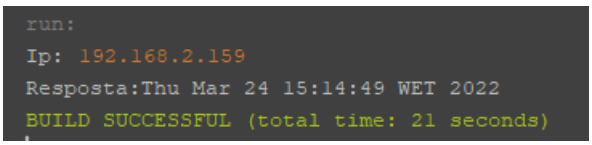

Figura 1 - Comunicação com server de um colega

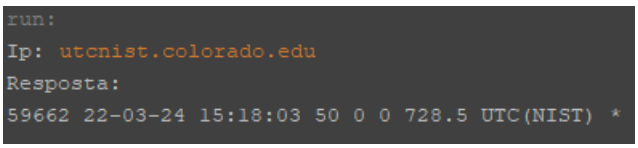

Figura 2 - Comunicação com server exterior

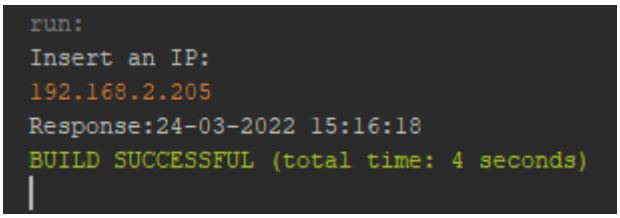

Figura 3 - Colega a comunicar com o meu servidor

Como desenvolvi o código:

Como porta para o cliente e servidor usa-se a porta 13 que é a porta que se usa em servidores daytime.
Para o cliente conseguir comunicar com servers externos ou o server interno (localhost) este usa o formato ASCII pois é de acordo a especificação RFC 867 e os servers daytime também usam esse mesmo formato.

2. Exercício 2

Execução do programa:

```
run:
Ip: localhost
29-03-2022 16:29:44
Mensagem: ola
Resposta: invalid command
Mensagem: datetime
Resposta: 29-03-2022 16:29:57
Mensagem: date
Resposta: 29-03-2022
Mensagem: time
Resposta: 16:30:00
Mensagem: help
Resposta: datetime/date/time
Mensagem: end
A encerrar...
BUILD SUCCESSFUL (total time: 26 seconds)
```

Figura 4 - O meu programa Cliente a comunicar com o programa Servidor

```
run:
Received command: ola Local IP: 127.0.0.1 Local Port: 13 Remote IP: 127.0.0.1 Remote Port: 65330

Received command: datetime Local IP: 127.0.0.1 Local Port: 13 Remote IP: 127.0.0.1 Remote Port: 65330

Received command: date Local IP: 127.0.0.1 Local Port: 13 Remote IP: 127.0.0.1 Remote Port: 65330

Received command: time Local IP: 127.0.0.1 Local Port: 13 Remote IP: 127.0.0.1 Remote Port: 65330

Received command: help Local IP: 127.0.0.1 Local Port: 13 Remote IP: 127.0.0.1 Remote Port: 65330

Received command: end Local IP: 127.0.0.1 Local Port: 13 Remote IP: 127.0.0.1 Remote Port: 65330
```

Figura 5 - O meu programa Servidor a receber as mensagens do programa Cliente

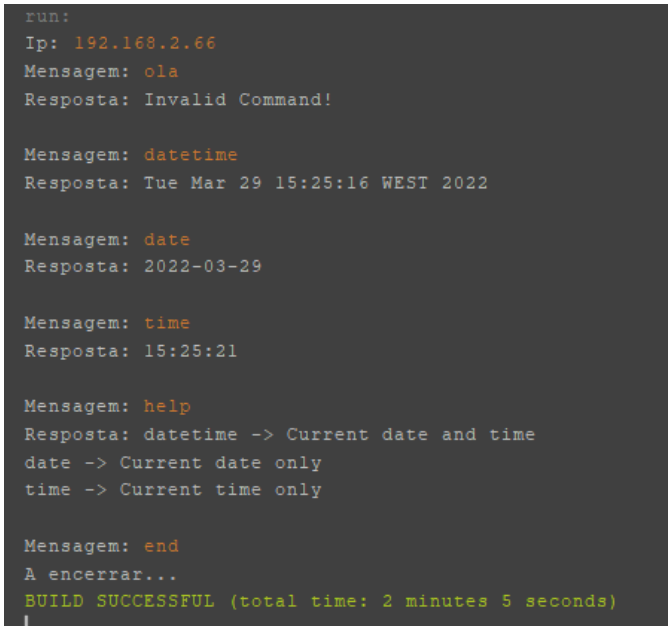

Figura 6 - O meu Cliente a comunicar com servidor de colega

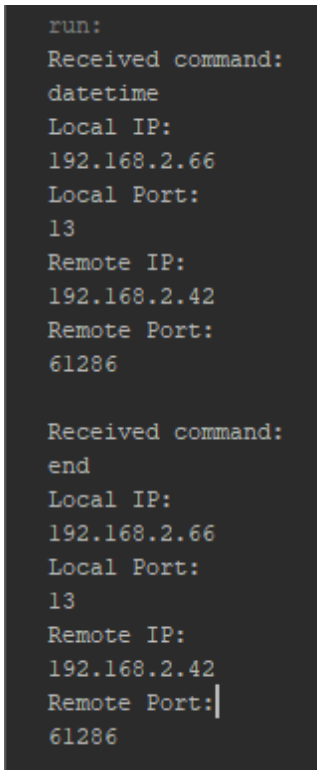

Figura 7 - Servidor de colega a receber as minhas mensagens

Nota: Apenas mostra duas mensagens pois esta imagem é editada e na verdade no servidor do colega aparece todas as mensagens que enviei.

```
Received command: ola Local IP: 192.168.2.42 Local Port: 13 Remote IP: 192.168.2.66 Remote Port: 52335

Received command: datetime Local IP: 192.168.2.42 Local Port: 13 Remote IP: 192.168.2.66 Remote Port: 52335

Received command: date Local IP: 192.168.2.42 Local Port: 13 Remote IP: 192.168.2.66 Remote Port: 52335

Received command: time Local IP: 192.168.2.42 Local Port: 13 Remote IP: 192.168.2.66 Remote Port: 52335

Received command: help Local IP: 192.168.2.42 Local Port: 13 Remote IP: 192.168.2.66 Remote Port: 52335

Received command: end Local IP: 192.168.2.42 Local Port: 13 Remote IP: 192.168.2.66 Remote Port: 52335
```

Figura 8 - O meu servidor a receber os comandos do cliente de um colega

```
run:
Insert an IP:
192.168.2.42
Insert the message to send:
ola
Answer:
invalid command
Insert the message to send:
datetime
Answer:
29-03-2022 15:28:33
Insert the message to send:
date
Answer:
29-03-2022
Insert the message to send:
time
Answer:
15:28:39
Insert the message to send:
help
Answer:
datetime/date/time
Insert the message to send:
end
BUILD SUCCESSFUL (total time: 19 seconds)
```

Figura 9 - Cliente de um colega a enviar mensagens ao meu servidor

Como desenvolvi o código:

```
String ip = inputString("Ip: ");
Socket socket = new Socket(ip, 13);
String answer;

DataOutputStream output = new DataOutputStream(socket.getOutputStream());
DataInputStream input = new DataInputStream(socket.getInputStream());

answer = input.readUTF();
System.out.println(answer);

while(true){

String s = inputString("Mensagem: ");

if (s.equals("end")) {
    output.writeUTF("end");
    break;
}

output.writeUTF(s);

answer = input.readUTF();
System.out.println("Resposta: " + answer);
}

//answer = input.readUTF();
//System.out.println(answer);
input.close();
output.close();
socket.close();
System.out.println("A encerrar...");
```

Figura 10 - Programa Cliente

O programa começa por perguntar o ip que o Cliente deseja conectar-se.

Depois crio o output e o input do programa que vai servir para o cliente poder enviar e receber informação do servidor (o servidor também usa output/input).

O Cliente começa por receber uma datetime do servidor antes de começar a mandar mensagens.

Ao entrar no while o programa Cliente pergunta ao utilizador a mensagem que deseja enviar para o servidor (output), depois proceda a receber a mensagem de resposta do servidor (input).

Se o utilizador desejar sair pode escrever “end” e o programa procede a enviar uma mensagem “end” ao servidor para o servidor fechar a socket atual e o cliente fecha a sua socket simultaneamente.

```
while (true) {

    command = "";
    socket = serversocket.accept();
    output = new DataOutputStream(socket.getOutputStream());
    input = new DataInputStream(socket.getInputStream());

    currentDateTime = LocalDateTime.now();
    formatter = DateTimeFormatter.ofPattern("dd-MM-yyyy HH:mm:ss");
    t = currentDateTime.format(formatter);
    output.writeUTF(t);

    while (!command.equals("end")) {

        command = input.readUTF();

        currentDateTime = LocalDateTime.now();

        System.out.println("Received command: " + command + " Local IP: " + socket.getLocalAddress().getHostAddress()
                + " Local Port: " + socket.getLocalPort()
                + " Remote IP: " + socket.getInetAddress().getHostAddress()
                + " Remote Port: " + socket.getPort() + "\n");

        switch (command) {
            case "datetime": {
                formatter = DateTimeFormatter.ofPattern("dd-MM-yyyy HH:mm:ss");
                t = currentDateTime.format(formatter);
                output.writeUTF(t);
                break;
            }
```

Figura 11 - Programa Servidor (Conexão e Switch)

```
            default:
                t = "invalid command";
                output.writeUTF(t);
        }

    }

    input.close();
    output.close();
    socket.close();
```

Figura 12 - Programa Servidor (fecho da socket, input e output)

O programa Servidor contem dois ciclos. O ciclo externo estará sempre a funcionar pois o servidor não deve fechar. Nesse mesmo ciclo, usamos a função serversocket.accept() para conectar o Cliente, e antes de começar a receber mensagens do Cliente, o Servidor envia o datetime atual. No final desse ciclo fechamos a socket atual para fechar a ligação com o Cliente.

O ciclo interno recebe a mensagem que o Cliente envia e dá print no terminal da mensagem. No terminal escreve a mensagem que recebeu, o local ip e port e o remote ip e port da mensagem. Depois entra no switch onde se a mensagem não for válida o servidor retorna “invalid command”, senão o servidor retorna ou “datetime” ou “date” ou “time”.

O ciclo interno sai quando o Cliente envia a mensagem de “end”.

Pergunta: No seu programa, enquanto um cliente está a ser atendido é possível a outro cliente efetuar uma nova ligação? Explicar o motivo.

Não, pois o servidor até fechar a sua socket não conseguirá aceitar outro cliente.

```
socket = serversocket.accept();
```

Figura 13 - Função que aceita um cliente se conectar

Esta função é executada ao início do programa e só volta a executar todas as vezes que um cliente se desconecta de um servidor.