

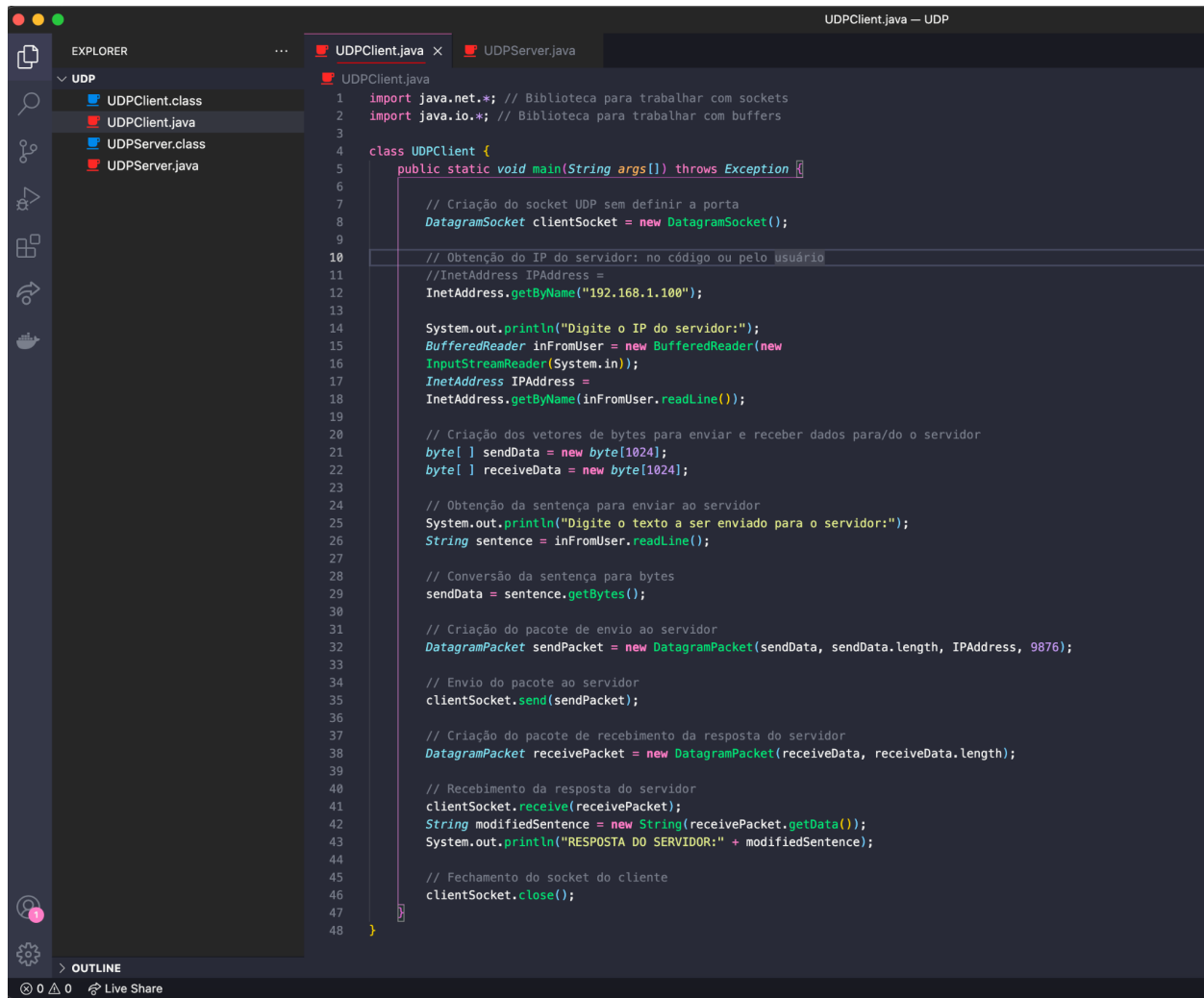
Laboratório de Redes de Computadores

2ª Prática

Bruna Gomes Camilo e Lorena Gomes de Oliveira Cabral

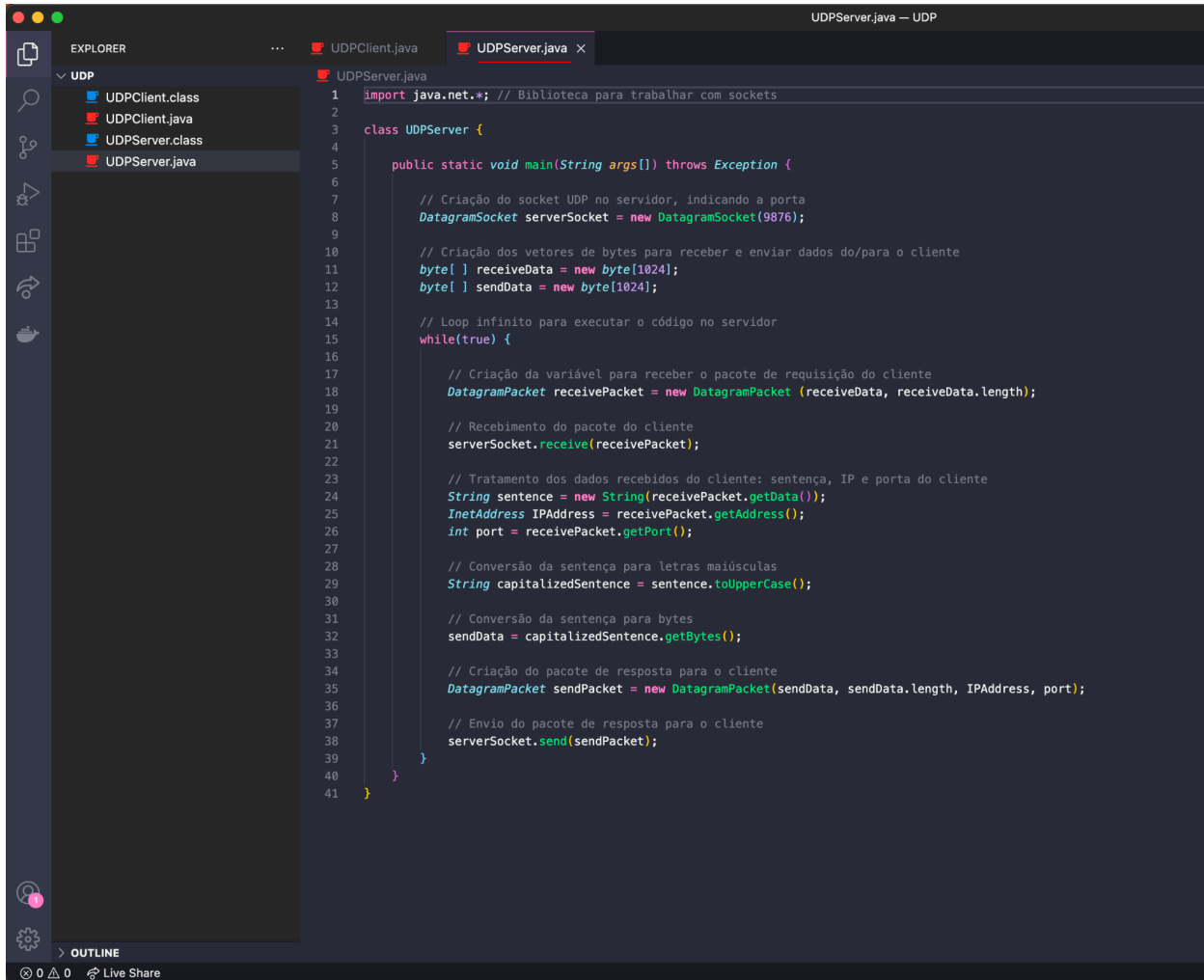
5 de abril de 2022

Questão 1



```
1 import java.net.*; // Biblioteca para trabalhar com sockets
2 import java.io.*; // Biblioteca para trabalhar com buffers
3
4 class UDPClient {
5     public static void main(String args[]) throws Exception {
6
7         // Criação do socket UDP sem definir a porta
8         DatagramSocket clientSocket = new DatagramSocket();
9
10        // Obtenção do IP do servidor: no código ou pelo usuário
11        InetAddress IPAddress =
12        InetAddress.getByName("192.168.1.100");
13
14        System.out.println("Digite o IP do servidor:");
15        BufferedReader inFromUser = new BufferedReader(new
16        InputStreamReader(System.in));
17        InetAddress IPAddress =
18        InetAddress.getByName(inFromUser.readLine());
19
20        // Criação dos vetores de bytes para enviar e receber dados para/do o servidor
21        byte[] sendData = new byte[1024];
22        byte[] receiveData = new byte[1024];
23
24        // Obtenção da sentença para enviar ao servidor
25        System.out.println("Digite o texto a ser enviado para o servidor:");
26        String sentence = inFromUser.readLine();
27
28        // Conversão da sentença para bytes
29        sendData = sentence.getBytes();
30
31        // Criação do pacote de envio ao servidor
32        DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length, IPAddress, 9876);
33
34        // Envio do pacote ao servidor
35        clientSocket.send(sendPacket);
36
37        // Criação do pacote de recebimento da resposta do servidor
38        DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);
39
40        // Recebimento da resposta do servidor
41        clientSocket.receive(receivePacket);
42        String modifiedSentence = new String(receivePacket.getData());
43        System.out.println("RESPOSTA DO SERVIDOR:" + modifiedSentence);
44
45        // Fechamento do socket do cliente
46        clientSocket.close();
47    }
48 }
```

Questão 2



```
1 import java.net.*; // Biblioteca para trabalhar com sockets
2
3 class UDPServer {
4
5     public static void main(String args[]) throws Exception {
6
7         // Criação do socket UDP no servidor, indicando a porta
8         DatagramSocket serverSocket = new DatagramSocket(9876);
9
10        // Criação dos vetores de bytes para receber e enviar dados do/para o cliente
11        byte[] receiveData = new byte[1024];
12        byte[] sendData = new byte[1024];
13
14        // Loop infinito para executar o código no servidor
15        while(true) {
16
17            // Criação da variável para receber o pacote de requisição do cliente
18            DatagramPacket receivePacket = new DatagramPacket (receiveData, receiveData.length);
19
20            // Recebimento do pacote do cliente
21            serverSocket.receive(receivePacket);
22
23            // Tratamento dos dados recebidos do cliente: sentença, IP e porta do cliente
24            String sentence = new String(receivePacket.getData());
25            InetAddress IPAddress = receivePacket.getAddress();
26            int port = receivePacket.getPort();
27
28            // Conversão da sentença para letras maiúsculas
29            String capitalizedSentence = sentence.toUpperCase();
30
31            // Conversão da sentença para bytes
32            sendData = capitalizedSentence.getBytes();
33
34            // Criação do pacote de resposta para o cliente
35            DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length, IPAddress, port);
36
37            // Envio do pacote de resposta para o cliente
38            serverSocket.send(sendPacket);
39        }
40    }
41 }
```

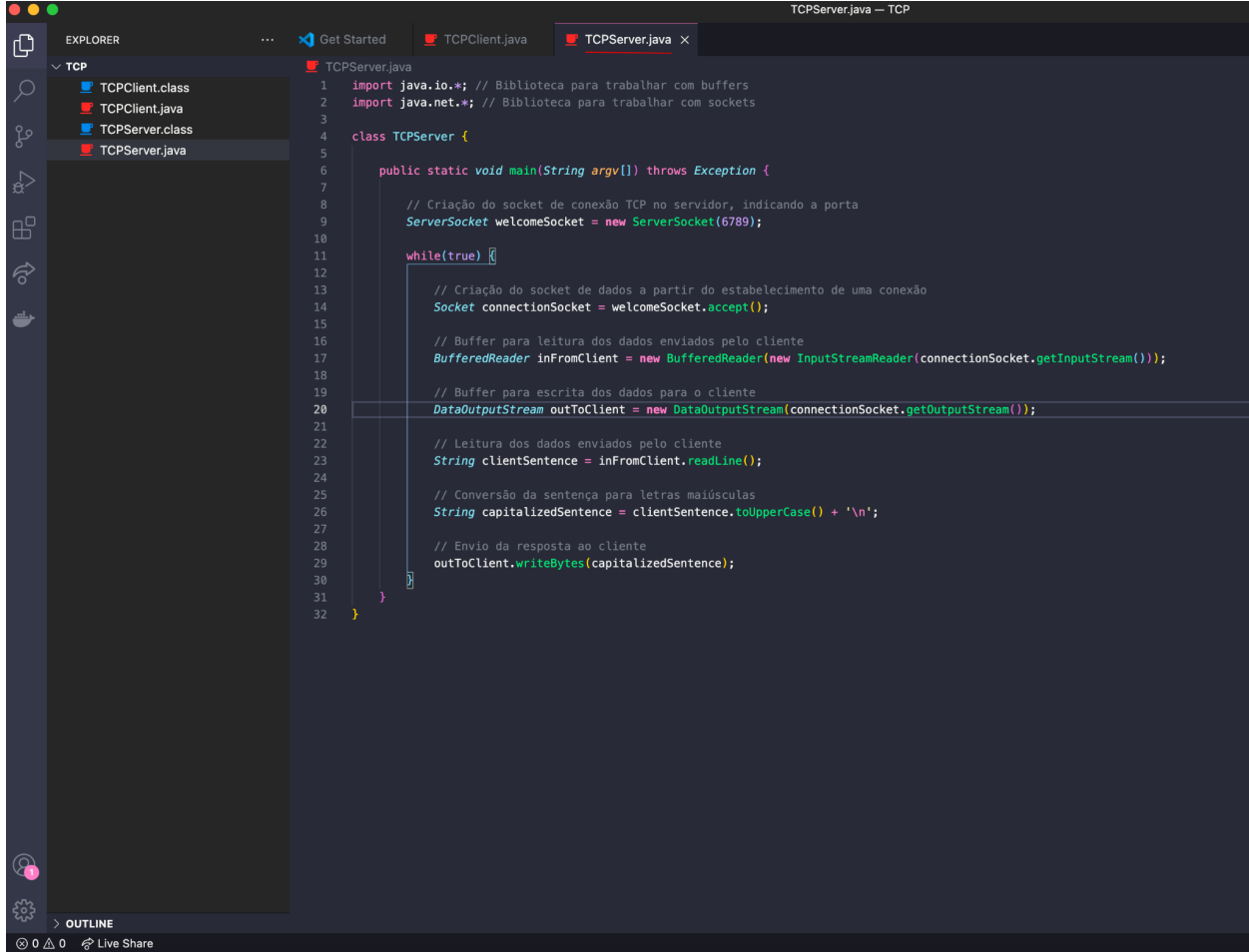
10/10/2019



Questão 4

```
1  import java.io.*; // Biblioteca para trabalhar com buffers
2  import java.net.*; // Biblioteca para trabalhar com sockets
3
4  class TCPClient {
5
6      public static void main(String argv[ ]) throws Exception {
7          // Criação do socket TCP do cliente, indicando o IP e a porta do servidor
8          Socket clientSocket = new Socket("127.0.0.1", 6789);
9
10         // Buffer para escrita dos dados para o servidor
11         DataOutputStream outToServer = new
12             DataOutputStream(clientSocket.getOutputStream());
13
14         // Buffer para leitura dos dados enviados pelo servidor
15         BufferedReader inFromServer = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
16
17         // Obtenção da sentença para enviar ao servidor
18         BufferedReader inFromUser = new BufferedReader(new InputStreamReader(System.in));
19         System.out.println("Digite o texto a ser enviado para o servidor:");
20         String sentence = inFromUser.readLine();
21
22         // Envio da sentença ao servidor
23         outToServer.writeBytes(sentence + '\n');
24
25         // Leitura da resposta enviada pelo servidor
26         String modifiedSentence = inFromServer.readLine();
27         System.out.println("RESPOSTA DO SERVIDOR: " + modifiedSentence);
28
29         // Fechamento do socket do cliente
30         clientSocket.close();
31     }
32 }
```

Questão 5



The screenshot shows a code editor with a dark theme. On the left, the 'EXPLORER' sidebar shows a project named 'TCP' containing four files: 'TCPClient.class', 'TCPClient.java', 'TCPServer.class', and 'TCPServer.java'. The 'TCPServer.java' file is selected and open in the main editor. The code in the editor is as follows:

```
1  import java.io.*; // Biblioteca para trabalhar com buffers
2  import java.net.*; // Biblioteca para trabalhar com sockets
3
4  class TCPServer {
5
6      public static void main(String argv[]) throws Exception {
7
8          // Criação do socket de conexão TCP no servidor, indicando a porta
9          ServerSocket welcomeSocket = new ServerSocket(6789);
10
11         while(true) {
12
13             // Criação do socket de dados a partir do estabelecimento de uma conexão
14             Socket connectionSocket = welcomeSocket.accept();
15
16             // Buffer para leitura dos dados enviados pelo cliente
17             BufferedReader inFromClient = new BufferedReader(new InputStreamReader(connectionSocket.getInputStream()));
18
19             // Buffer para escrita dos dados para o cliente
20             DataOutputStream outToClient = new DataOutputStream(connectionSocket.getOutputStream());
21
22             // Leitura dos dados enviados pelo cliente
23             String clientSentence = inFromClient.readLine();
24
25             // Conversão da sentença para letras maiúsculas
26             String capitalizedSentence = clientSentence.toUpperCase() + '\n';
27
28             // Envio da resposta ao cliente
29             outToClient.writeBytes(capitalizedSentence);
30
31         }
32     }
```

At the bottom of the editor, there is an 'OUTLINE' tab and a status bar showing '0 0 0' and a 'Live Share' icon.

Questão 6

The screenshot shows an IDE with two Java files: `TCPClient.java` and `TCPServer.java`. The `TCPClient.java` file contains the following code:

```
1 import java.io.*; // Biblioteca para trabalhar com buffers
2 import java.net.*; // Biblioteca para trabalhar com sockets
3
4 class TCPClient {
5
6
7     public static void main(String argv[]) throws Exception {
8         // Criação do socket TCP do cliente, indicando o IP e a porta do servidor
9         Socket clientSocket = new Socket("127.0.0.1", 6789);
10
11         // Buffer para escrita dos dados para o servidor
12         DataOutputStream outToServer = new
```

Below the IDE, there are two terminal windows. The left terminal window shows the following commands and output:

```
Last login: Tue Apr 5 19:46:38 on ttys002
+ ~ cd Documents
+ Documents ls
+ Documents cd TCP
+ TCP javac TCPServer.java
+ TCP java TCPServer
```

The right terminal window shows the following commands and output:

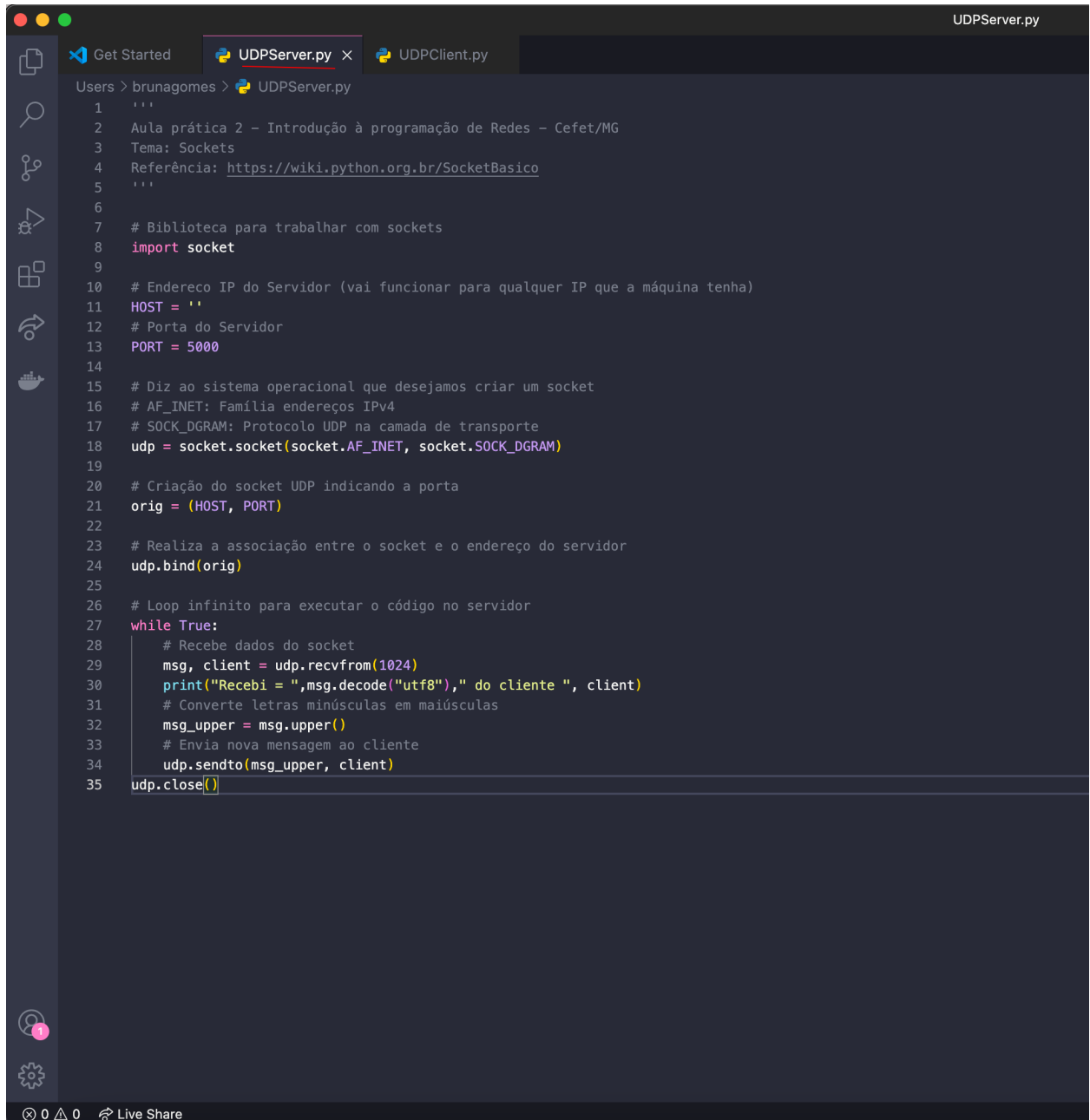
```
brunagomes@ip-172-16-2-67:~/Documents/TCP
+ ~ cd Documents
+ Documents ls
+ Documents cd TCP
+ TCP javac TCPClient.java
+ TCP java TCPClient
Digite o texto a ser enviado para o servidor:
bruna e lorena
RESPOSTA DO SERVIDOR: BRUNA E LORENA
+ TCP []
```

Questão 7

1. UDPClient.py

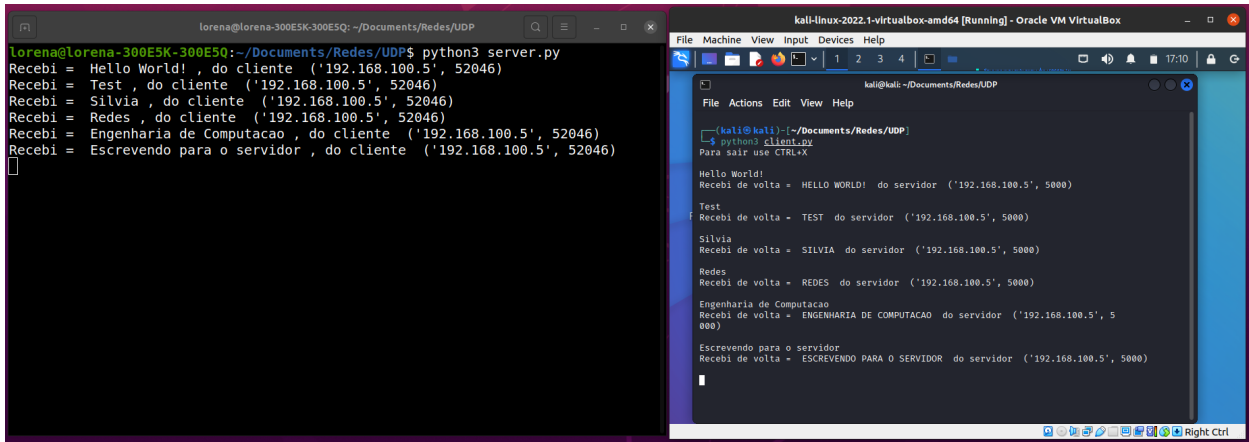
```
Users > brunagomes > UDPClient.py
1  '''
2  Aula prática 2 - Introdução à programação de Redes - Cefet/MG
3  Tema: Sockets
4  Referência: https://wiki.python.org.br/SocketBasico
5  '''
6
7  # Biblioteca para trabalhar com sockets
8  import socket
9
10 # Endereço IP do Servidor
11 HOST = '192.168.100.5'
12 # Porta do Servidor
13 PORT = 5000
14
15 # Diz ao sistema operacional que desejamos criar um socket
16 # AF_INET: Família endereços IPv4
17 # SOCK_DGRAM: Protocolo UDP na camada de transporte
18 udp = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
19
20 # Criação do socket UDP indicando a porta do servidor
21 dest = (HOST, PORT)
22
23 print("Para sair use CTRL+X\n")
24
25 # Recebe dados do teclado
26 msg = input()
27
28 # Enquanto não for usado CTRL+X
29 while msg != '\x18':
30     # Converte a mensagem para bytes e envia ao servidor
31     udp.sendto(bytes(msg,"utf8"), dest)
32     # Recebe nova mensagem do servidor
33     msg_upper, server = udp.recvfrom(1024)
34     print("Recebi de volta = ",msg_upper.decode("utf8")," do servidor ", server,"\n")
35     msg = input()
36 udp.close()
```


2. UDPServer.py



```
UDPServer.py
Users > brunagomes > UDPServer.py
1  '''
2  Aula prática 2 - Introdução à programação de Redes - Cefet/MG
3  Tema: Sockets
4  Referência: https://wiki.python.org.br/SocketBasico
5  '''
6
7  # Biblioteca para trabalhar com sockets
8  import socket
9
10 # Endereco IP do Servidor (vai funcionar para qualquer IP que a máquina tenha)
11 HOST = ''
12 # Porta do Servidor
13 PORT = 5000
14
15 # Diz ao sistema operacional que desejamos criar um socket
16 # AF_INET: Família endereços IPv4
17 # SOCK_DGRAM: Protocolo UDP na camada de transporte
18 udp = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
19
20 # Criação do socket UDP indicando a porta
21 orig = (HOST, PORT)
22
23 # Realiza a associação entre o socket e o endereço do servidor
24 udp.bind(orig)
25
26 # Loop infinito para executar o código no servidor
27 while True:
28     # Recebe dados do socket
29     msg, client = udp.recvfrom(1024)
30     print("Recebi = ",msg.decode("utf8")," do cliente ", client)
31     # Converte letras minúsculas em maiúsculas
32     msg_upper = msg.upper()
33     # Envia nova mensagem ao cliente
34     udp.sendto(msg_upper, client)
35 udp.close()
```

3. Resultado UDP



The image displays two terminal windows side-by-side, illustrating a UDP communication test. The left window, titled 'lorena@lorena-300E5K-300E5Q: ~/Documents/Redes/UDP', shows the execution of a Python server script. The output lists five received messages from a client at IP 192.168.100.5, port 52046: 'Hello World!', 'Test', 'Silvia', 'Redes', and 'Engenharia de Computacao'. The right window, titled 'kali@kali: ~/Documents/Redes/UDP', shows the execution of a Python client script. The output lists five sent messages to a server at IP 192.168.100.5, port 5000: 'Hello World!', 'Test', 'Silvia', 'Redes', and 'Engenharia de Computacao'. Each message is followed by a confirmation line: 'Recebi de volta = [message] do servidor ('192.168.100.5', 5000)'.

```
lorena@lorena-300E5K-300E5Q: ~/Documents/Redes/UDP$ python3 server.py
Recebi = Hello World! , do cliente ('192.168.100.5', 52046)
Recebi = Test , do cliente ('192.168.100.5', 52046)
Recebi = Silvia , do cliente ('192.168.100.5', 52046)
Recebi = Redes , do cliente ('192.168.100.5', 52046)
Recebi = Engenharia de Computacao , do cliente ('192.168.100.5', 52046)
Recebi = Escrevendo para o servidor , do cliente ('192.168.100.5', 52046)

```

```
kali@kali: ~/Documents/Redes/UDP$ python3 client.py
Para sair use CTRL+X

Hello World!
Recebi de volta = HELLO WORLD! do servidor ('192.168.100.5', 5000)

Test
Recebi de volta = TEST do servidor ('192.168.100.5', 5000)

Silvia
Recebi de volta = SILVIA do servidor ('192.168.100.5', 5000)

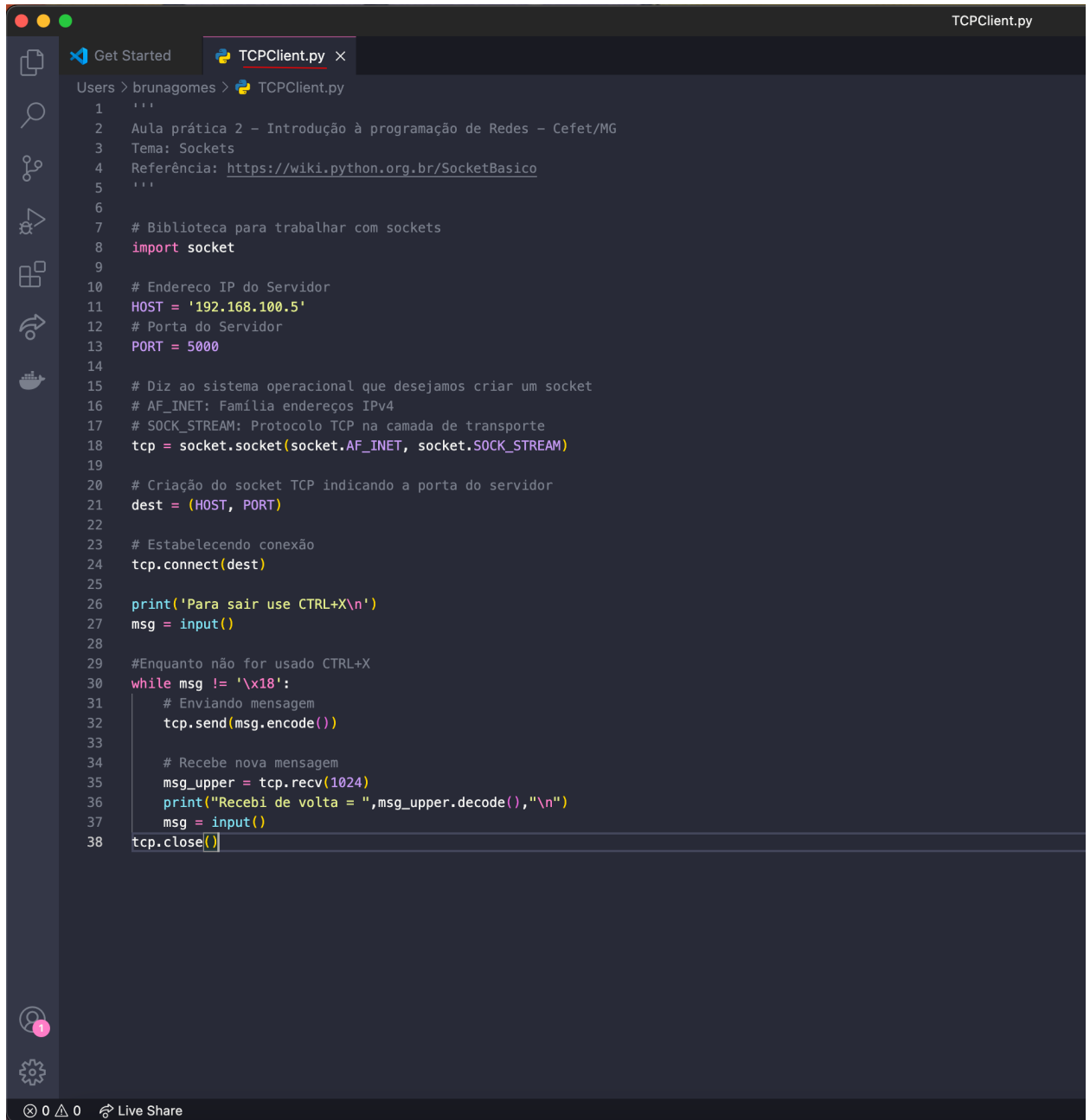
Redes
Recebi de volta = REDES do servidor ('192.168.100.5', 5000)

Engenharia de Computacao
Recebi de volta = ENGENHARIA DE COMPUTACAO do servidor ('192.168.100.5', 5000)

Escrevendo para o servidor
Recebi de volta = ESCRREVENDO PARA O SERVIDOR do servidor ('192.168.100.5', 5000)

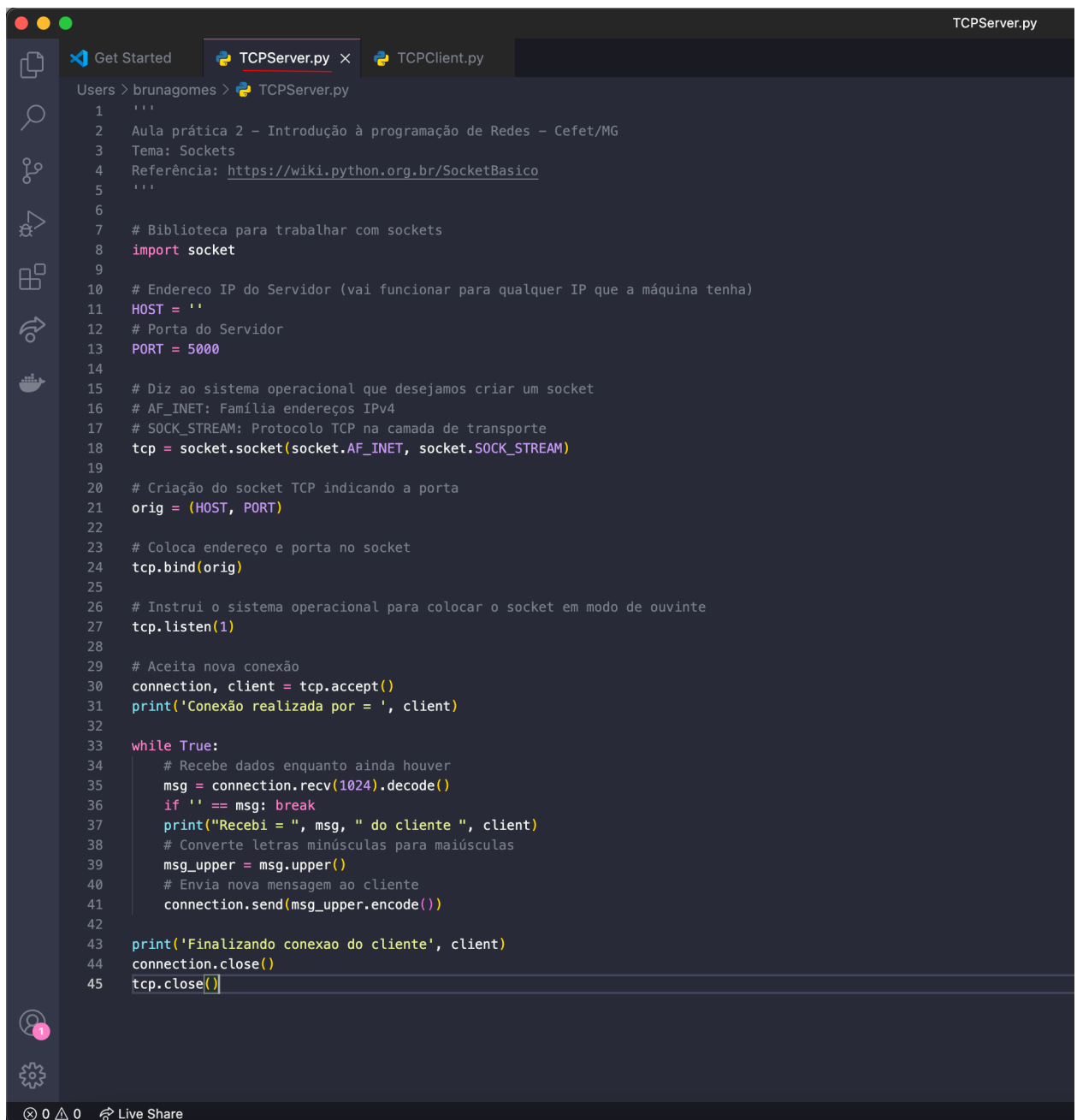
```

4. TCPClient.py



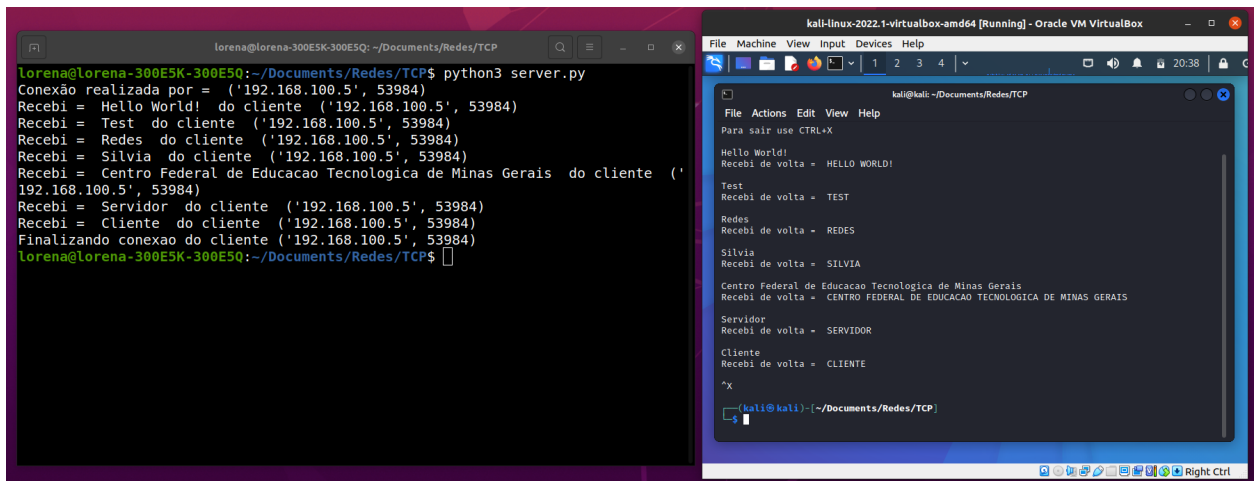
```
Users > brunagomes > TCPClient.py
1  '''
2  Aula prática 2 - Introdução à programação de Redes - Cefet/MG
3  Tema: Sockets
4  Referência: https://wiki.python.org.br/SocketBasico
5  '''
6
7  # Biblioteca para trabalhar com sockets
8  import socket
9
10 # Endereço IP do Servidor
11 HOST = '192.168.100.5'
12 # Porta do Servidor
13 PORT = 5000
14
15 # Diz ao sistema operacional que desejamos criar um socket
16 # AF_INET: Família endereços IPv4
17 # SOCK_STREAM: Protocolo TCP na camada de transporte
18 tcp = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
19
20 # Criação do socket TCP indicando a porta do servidor
21 dest = (HOST, PORT)
22
23 # Estabelecendo conexão
24 tcp.connect(dest)
25
26 print('Para sair use CTRL+X\n')
27 msg = input()
28
29 # Enquanto não for usado CTRL+X
30 while msg != '\x18':
31     # Enviando mensagem
32     tcp.send(msg.encode())
33
34     # Recebe nova mensagem
35     msg_upper = tcp.recv(1024)
36     print("Recebi de volta = ", msg_upper.decode(), "\n")
37     msg = input()
38 tcp.close()
```

5. TCPServer.py



```
TCPServer.py
Users > brunagomes > TCPServer.py
1  '''
2  Aula prática 2 - Introdução à programação de Redes - Cefet/MG
3  Tema: Sockets
4  Referência: https://wiki.python.org.br/SocketBasico
5  '''
6
7  # Biblioteca para trabalhar com sockets
8  import socket
9
10 # Endereço IP do Servidor (vai funcionar para qualquer IP que a máquina tenha)
11 HOST = ''
12 # Porta do Servidor
13 PORT = 5000
14
15 # Diz ao sistema operacional que desejamos criar um socket
16 # AF_INET: Família endereços IPv4
17 # SOCK_STREAM: Protocolo TCP na camada de transporte
18 tcp = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
19
20 # Criação do socket TCP indicando a porta
21 orig = (HOST, PORT)
22
23 # Coloca endereço e porta no socket
24 tcp.bind(orig)
25
26 # Instrui o sistema operacional para colocar o socket em modo de ouvinte
27 tcp.listen(1)
28
29 # Aceita nova conexão
30 connection, client = tcp.accept()
31 print('Conexão realizada por = ', client)
32
33 while True:
34     # Recebe dados enquanto ainda houver
35     msg = connection.recv(1024).decode()
36     if '' == msg: break
37     print("Recebi = ", msg, " do cliente ", client)
38     # Converte letras minúsculas para maiúsculas
39     msg_upper = msg.upper()
40     # Envia nova mensagem ao cliente
41     connection.send(msg_upper.encode())
42
43 print('Finalizando conexão do cliente', client)
44 connection.close()
45 tcp.close()
```

6. Resultado TCP



The image displays two terminal windows side-by-side, illustrating a TCP communication. The left window, titled 'lorena@lorena-300E5K-300E5Q: ~/Documents/Redes/TCP', shows the execution of a Python script 'server.py'. The script prints a series of messages received from a client, including 'Hello World!', 'Test', 'Redes', 'Silvia', 'Centro Federal de Educacao Tecnologica de Minas Gerais', 'Servidor', and 'Cliente'. The right window, titled 'kali@kali: ~/Documents/Redes/TCP', shows the output of a client program. It displays the same messages received from the server, including 'Hello World!', 'Test', 'Redes', 'Silvia', 'Centro Federal de Educacao Tecnologica de Minas Gerais', 'Servidor', and 'Cliente'. The client window also shows a prompt for the user to press Ctrl+C to exit.

```
lorena@lorena-300E5K-300E5Q: ~/Documents/Redes/TCP$ python3 server.py
Conexão realizada por = ('192.168.100.5', 53984)
Recebi = Hello World! do cliente ('192.168.100.5', 53984)
Recebi = Test do cliente ('192.168.100.5', 53984)
Recebi = Redes do cliente ('192.168.100.5', 53984)
Recebi = Silvia do cliente ('192.168.100.5', 53984)
Recebi = Centro Federal de Educacao Tecnologica de Minas Gerais do cliente ('192.168.100.5', 53984)
Recebi = Servidor do cliente ('192.168.100.5', 53984)
Recebi = Cliente do cliente ('192.168.100.5', 53984)
Finalizando conexao do cliente ('192.168.100.5', 53984)
lorena@lorena-300E5K-300E5Q:~/Documents/Redes/TCP$
```

```
kali@kali: ~/Documents/Redes/TCP$
File Actions Edit View Help
Para sair use CTRL+X
Hello World!
Recebi de volta = HELLO WORLD!
Test
Recebi de volta = TEST
Redes
Recebi de volta = REDES
Silvia
Recebi de volta = SILVIA
Centro Federal de Educacao Tecnologica de Minas Gerais
Recebi de volta = CENTRO FEDERAL DE EDUCACAO TECNOLOGICA DE MINAS GERAIS
Servidor
Recebi de volta = SERVIDOR
Cliente
Recebi de volta = CLIENTE
^X
kali@kali:~/Documents/Redes/TCP$
```