

- 1) Os programas a seguir estão disponíveis no endereço [www.cdk5.net/ipc]. Utilize-os para fazer uma série de testes para determinar as condições nas quais os datagramas, às vezes, são descartados. Dica: o programa cliente deve ser capaz de variar o número de mensagens enviadas e seus tamanhos; o servidor deve detectar quando uma mensagem de um cliente específico é perdida.

Foi utilizado o código UDP e foi observado que ao tentar um valor demasiadamente grande no for de concatenação da string o Java dava o erro de heap space, segue o código utilizado:

Pequeno valor para concatenação

[illegible]

Valor maior para concatenação

The screenshot displays an IDE with a Java file named `UDPServer.java` and its execution output in a terminal window.

UDPServer.java Code:

```

1  import java.net.*;
2  import java.io.*;
3  public class UDPServer{
4      public static void main(String args[]){
5          DatagramSocket aSocket = null;
6          try{
7              aSocket = new DatagramSocket(port:6789);
8              // create socket at agreed port
9              byte[] buffer = new byte[1000];
10             while(true){
11                 DatagramPacket request = new DatagramPacket(buffer, buffer.length);
12                 aSocket.receive(request);
13                 DatagramPacket reply = new DatagramPacket(request.getData(), request.getLength());
14                 request.getAddress(), request.getPort());
15                 String atividade = reply.getData().toString();
16                 for (int i = 0; i < 109; i++) {
17                     atividade += atividade;
18                 }
19                 reply.setData(atividade.getBytes(), offset:0, atividade.length());
20                 aSocket.send(reply);
21             }
22         }catch (SocketException e){System.out.println("Socket: " + e.getMessage());}
23         }catch (IOException e) {System.out.println("IO: " + e.getMessage());}
24         }finally {if(aSocket != null) aSocket.close();}
25     }
26 }
27
28

```

Terminal Output:

```

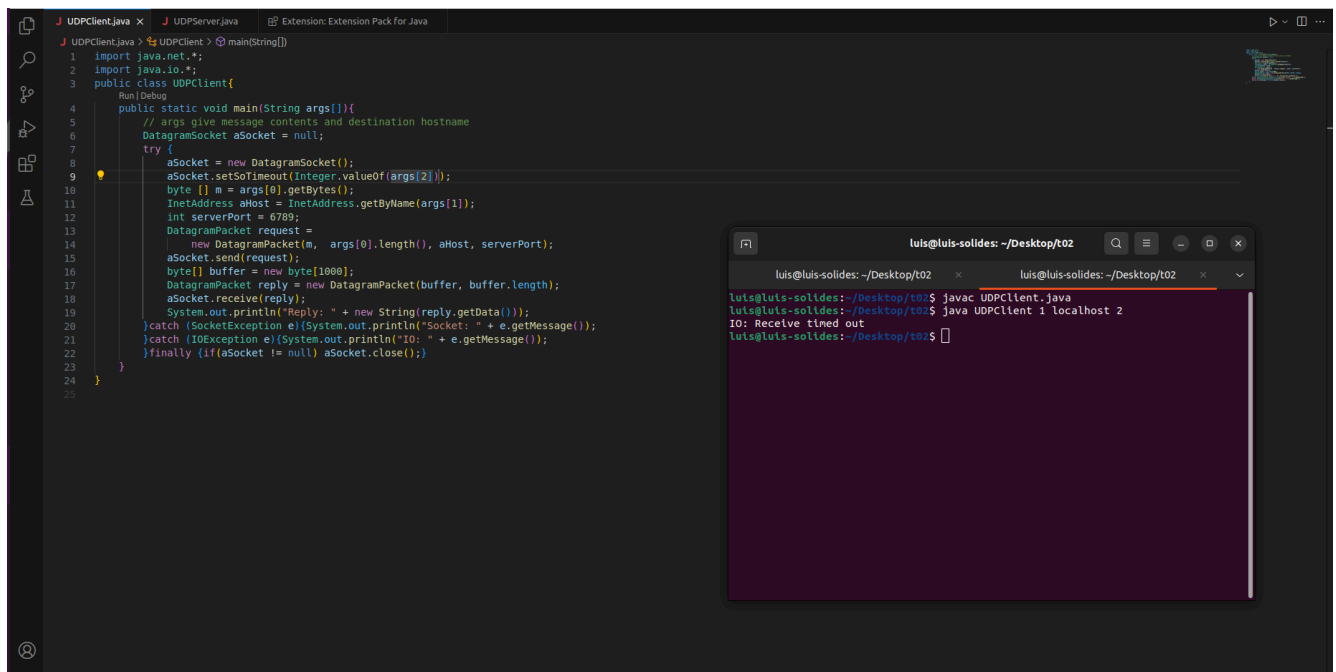
luis@luis-solides: ~/Desktop/t02
luis@luis-solides: ~/Desktop/t02 $ javac UDPServer.java
luis@luis-solides: ~/Desktop/t02 $ java UDPServer
IO: Message too long (sendto failed)
IO: Message too long (sendto failed)
luis@luis-solides: ~/Desktop/t02 $ java UDPServer
Exception in thread "main" java.lang.OutOfMemoryError: Java heap space
at java.util.Arrays.copyOf(Arrays.java:3332)
at java.lang.AbstractStringBuilder.ensureCapacityInternal(AbstractString
Builder.java:124)
at java.lang.AbstractStringBuilder.append(AbstractStringBuilder.java:448)
at java.lang.StringBuilder.append(StringBuilder.java:136)
at UDPServer.main(UDPServer.java:18)
luis@luis-solides: ~/Desktop/t02 $

```

- 2) Use o programa cliente e servidor UDP do exercício anterior para fazer um programa cliente que leia repetidamente uma linha de entrada do usuário, a envie para o servidor em uma mensagem

datagrama UDP e receba uma mensagem do servidor. O cliente estabelece um tempo limite em seu soquete para que possa informar o usuário quando o servidor não responder.

Basta utilizar o método `setSoTimeout` no objeto `aSocket`



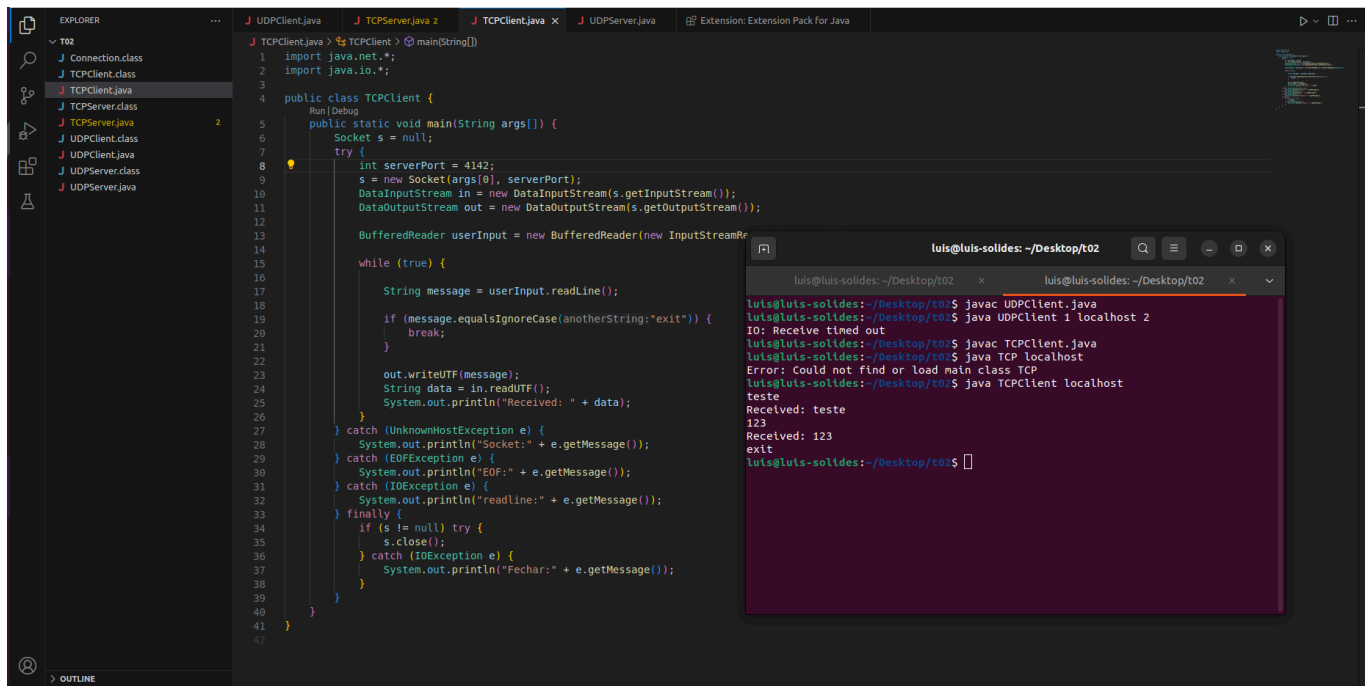
The screenshot shows an IDE with a file named `UDPCClient.java` open. The code is as follows:

```
1 import java.net.*;
2 import java.io.*;
3 public class UDPCClient {
4     public static void main(String args[]) {
5         // args give message contents and destination hostname
6         DatagramSocket aSocket = null;
7         try {
8             aSocket = new DatagramSocket();
9             aSocket.setSoTimeout(Integer.valueOf(args[2]));
10            byte[] m = args[0].getBytes();
11            InetAddress aHost = InetAddress.getByName(args[1]);
12            int serverPort = 6789;
13            DatagramPacket request =
14                new DatagramPacket(m, args[0].length(), aHost, serverPort);
15            aSocket.send(request);
16            byte[] buffer = new byte[1000];
17            DatagramPacket reply = new DatagramPacket(buffer, buffer.length);
18            aSocket.receive(reply);
19            System.out.println("Reply: " + new String(reply.getData()));
20        } catch (SocketException e) { System.out.println("Socket: " + e.getMessage()); }
21        } catch (IOException e) { System.out.println("IO: " + e.getMessage()); }
22        } finally { if (aSocket != null) aSocket.close(); }
23    }
24 }
25 }
```

To the right, a terminal window shows the following commands and output:

```
luis@luis-solides: ~/Desktop/t02
luis@luis-solides:~/Desktop/t02$ javac UDPCClient.java
luis@luis-solides:~/Desktop/t02$ java UDPCClient 1 localhost 2
IO: Receive timed out
luis@luis-solides:~/Desktop/t02$
```

3) Modifique-os de modo que o cliente leia repetidamente uma linha de entrada do usuário e a escreva no fluxo. O servidor deve ler repetidamente o fluxo, imprimindo o resultado de cada leitura. Faça uma comparação entre o envio de dados em mensagens de datagrama UDP e por meio de um fluxo.



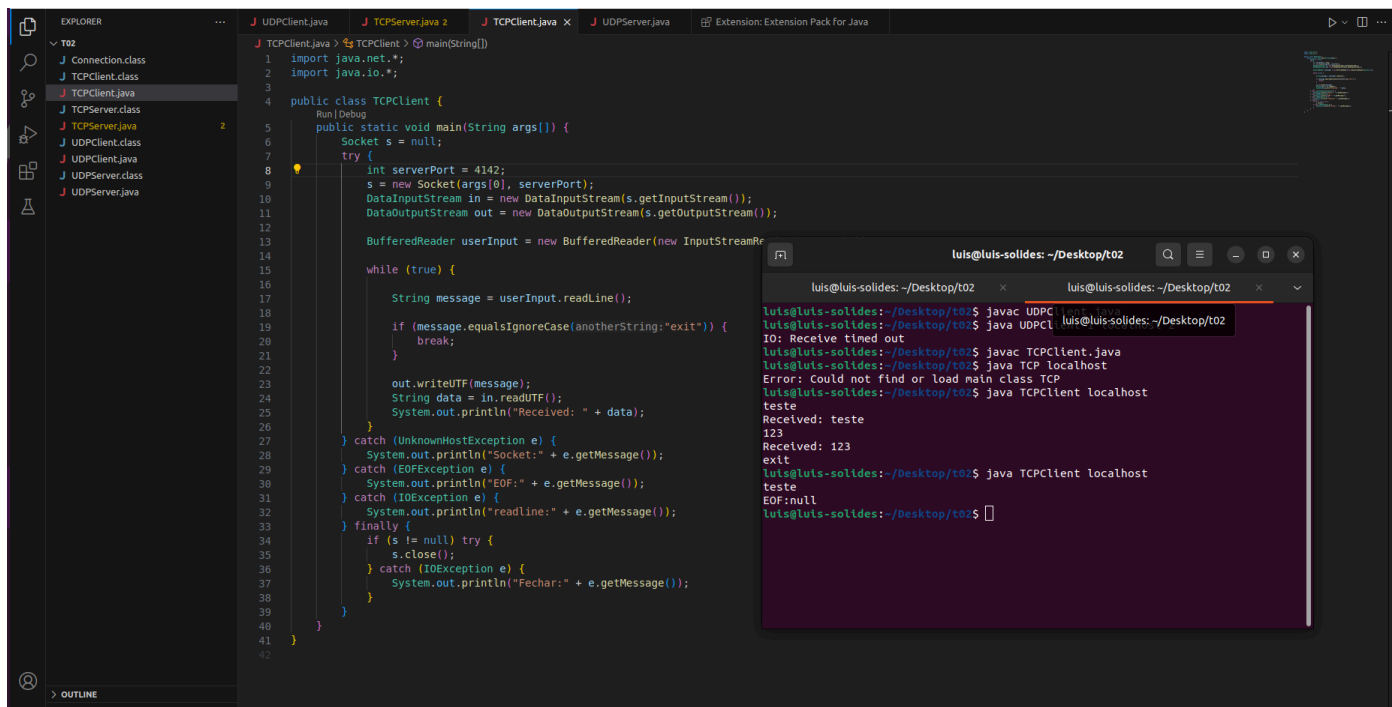
The screenshot shows an IDE with a file named `TCPClient.java` open. The code is as follows:

```
1 import java.net.*;
2 import java.io.*;
3
4 public class TCPClient {
5     public static void main(String args[]) {
6         Socket s = null;
7         try {
8             int serverPort = 4142;
9             s = new Socket(args[0], serverPort);
10            DataInputStream in = new DataInputStream(s.getInputStream());
11            DataOutputStream out = new DataOutputStream(s.getOutputStream());
12            BufferedReader userInput = new BufferedReader(new InputStreamReader(s.getInputStream()));
13
14            while (true) {
15                String message = userInput.readLine();
16
17                if (message.equalsIgnoreCase("exit")) {
18                    break;
19                }
20
21                out.writeUTF(message);
22                String data = in.readUTF();
23                System.out.println("Received: " + data);
24            }
25        } catch (UnknownHostException e) {
26            System.out.println("Socket: " + e.getMessage());
27        } catch (EOFException e) {
28            System.out.println("EOF: " + e.getMessage());
29        } catch (IOException e) {
30            System.out.println("readline: " + e.getMessage());
31        } finally {
32            if (s != null) try {
33                s.close();
34            } catch (IOException e) {
35                System.out.println("Fechar: " + e.getMessage());
36            }
37        }
38    }
39 }
40
41 }
42 }
```

To the right, a terminal window shows the following commands and output:

```
luis@luis-solides: ~/Desktop/t02
luis@luis-solides:~/Desktop/t02$ javac UDPCClient.java
luis@luis-solides:~/Desktop/t02$ java UDPCClient 1 localhost 2
IO: Receive timed out
luis@luis-solides:~/Desktop/t02$ javac TCPClient.java
luis@luis-solides:~/Desktop/t02$ java TCP localhost
Error: Could not find or load main class TCP
luis@luis-solides:~/Desktop/t02$ java TCPClient localhost
teste
Received: teste
123
Received: 123
exit
luis@luis-solides:~/Desktop/t02$
```

- 4) Use os programas desenvolvidos no exercício anterior para testar o efeito sobre o remetente quando o receptor falha e vice-versa (TCP).
Ao tentar enviar uma mensagem com o servidor fechado temos a seguinte resposta



The screenshot shows an IDE with a Java project named 'T02'. The file explorer on the left lists several files: 'Connection.class', 'TCPClient.class', 'TCPClient.java', 'TCPServer.class', 'TCPServer.java', 'UDPCClient.class', 'UDPCClient.java', 'UDPServer.class', and 'UDPServer.java'. The main editor displays the code for 'TCPClient.java', which is a Java class with a 'main' method. The code sets up a socket connection to a server at port 4142, reads input from the user, and sends it to the server. It also handles exceptions like 'UnknownHostException', 'EOFException', and 'IOException'. The terminal window on the right shows the execution of the program. It starts with the command 'javac UDPC...' and then 'java UDPC...'. The output shows the program running and receiving input from the user. The user enters 'teste' and the program prints 'Received: teste'. The user then enters 'exit' and the program prints 'Received: 123'. Finally, the user enters 'EOF:null' and the program prints 'EOF:null'.

```
1 import java.net.*;
2 import java.io.*;
3
4 public class TCPClient {
5     public static void main(String args[]) {
6         Socket s = null;
7         try {
8             int serverPort = 4142;
9             s = new Socket(args[0], serverPort);
10            DataInputStream in = new DataInputStream(s.getInputStream());
11            DataOutputStream out = new DataOutputStream(s.getOutputStream());
12
13            BufferedReader userInput = new BufferedReader(new InputStreamReader(System.in));
14
15            while (true) {
16                String message = userInput.readLine();
17
18                if (message.equalsIgnoreCase("exit")) {
19                    break;
20                }
21
22                out.writeUTF(message);
23                String data = in.readUTF();
24                System.out.println("Received: " + data);
25            }
26        } catch (UnknownHostException e) {
27            System.out.println("Socket: " + e.getMessage());
28        } catch (EOFException e) {
29            System.out.println("EOF: " + e.getMessage());
30        } catch (IOException e) {
31            System.out.println("readline: " + e.getMessage());
32        } finally {
33            if (s != null) try {
34                s.close();
35            } catch (IOException e) {
36                System.out.println("Fechar: " + e.getMessage());
37            }
38        }
39    }
40 }
```

```
luis@luis-solides: ~/Desktop/t02
luis@luis-solides: ~/Desktop/t02$ javac UDPC...
luis@luis-solides: ~/Desktop/t02$ java UDPC...
IO: Receive timed out
luis@luis-solides: ~/Desktop/t02$ javac TCPClient.java
luis@luis-solides: ~/Desktop/t02$ java TCP localhost
Error: Could not find or load main class TCP
luis@luis-solides: ~/Desktop/t02$ java TCPClient localhost
teste
Received: teste
123
Received: 123
exit
luis@luis-solides: ~/Desktop/t02$ java TCPClient localhost
teste
EOF:null
luis@luis-solides: ~/Desktop/t02$
```