

Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Aplicadas
Departamento de Computação e Sistemas

REDES DE COMPUTADORES 1

Primeiro Trabalho

Guilherme Marx Ferreira Tavares - 14.1.8006
Rafael Júnio Cota Cekiera - 14.2.5834
Professor - Theo Lins

João Monlevade
15 de novembro de 2017

Sumário

1	Introdução	2
2	Algoritmos	2
2.1	Servidor	2
2.2	Cliente	2
3	Apresentação e discussão dos resultados	3
3.1	Discussão dos resultados	11
4	Apêndice A	11
5	Bibliografia	14

1 Introdução

Neste trabalho prático, que tem a finalidade de aplicar o conceito de socket em redes de computadores, deve-se implementar uma comunicação entre processos situados em sistemas diferentes por meio do mecanismo socket.

2 Algoritmos

Um processo Cliente deve transmitir dois números e uma operação para o processo Servidor que, por sua vez, deve realizar a operação desejada entre os dois números e retornar ao cliente o resultado correto.

2.1 Servidor

- Primeiramente o servidor é iniciado na porta 55555, agora ele entra em estado de espera aguardando pela chamada da conexão do cliente.
- Uma vez recebida a solicitação de conexão, o servidor a aceita.
- O servidor recebe a mensagem do cliente contendo os dois valores e a operação por meio de uma string codificada em UTF, via TCP.
- O servidor decodifica a mensagem do cliente, colocando os valores em duas variáveis inteiras e fazendo seu calculo de acordo com o char contido no local da operação.
- Uma vez o resultado calculado, o servidor o coloca em uma nova String e a envia para o cliente novamente com codificação UTF com uso do protocolo TCP.
- O servidor encerra a conexão com o cliente e se fecha.

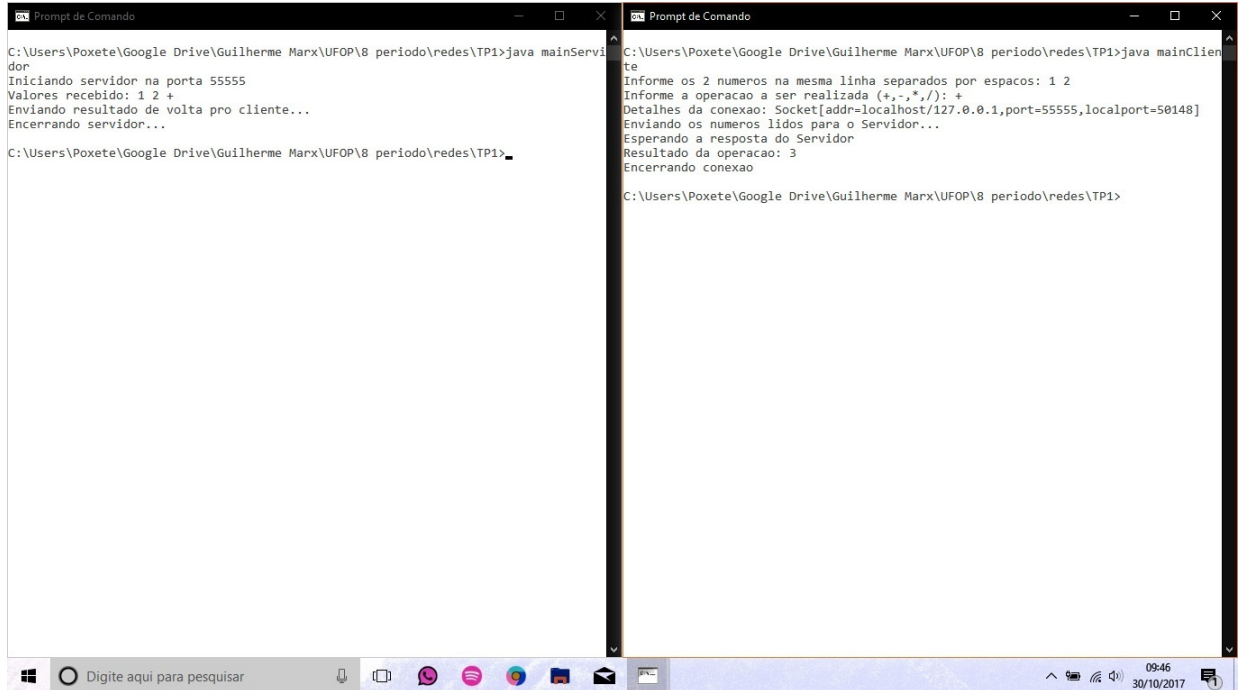
2.2 Cliente

- O cliente solicita os valores e a operação a ser realizada para o usuário e as salva em uma string.
- O cliente se conecta ao servidor na porta 55555 e envia ao servidor a string contendo os valores e a operação em formato UTF utilizando protocolo TCP.
- Cliente entra em modo de espera, aguardando a resposta do servidor.
- Ao obter a resposta do servidor, o cliente exibe essa resposta ao usuário, fecha sua conexão e finaliza.

3 Apresentação e discussão dos resultados

Foram realizado os testes de funcionamento do programa:

- Teste realizados:



```
C:\Users\Poxete\Google Drive\Guilherme Marx\UFOP\8 periodo\redes\TP1>java mainServidor
Iniciando servidor na porta 55555
Valores recebido: 1 2 +
Enviando resultado de volta pro cliente...
Encerrando servidor...
C:\Users\Poxete\Google Drive\Guilherme Marx\UFOP\8 periodo\redes\TP1>
```

```
C:\Users\Poxete\Google Drive\Guilherme Marx\UFOP\8 periodo\redes\TP1>java mainCliente
Informe os 2 numeros na mesma linha separados por espacos: 1 2
Informe a operacao a ser realizada (+,-,*,/): +
Detalhes da conexao: Socket[addr=localhost/127.0.0.1,port=55555,localport=50148]
Enviando os numeros lidos para o Servidor...
Esperando a resposta do Servidor
Resultado da operacao: 3
Encerrando conexao
C:\Users\Poxete\Google Drive\Guilherme Marx\UFOP\8 periodo\redes\TP1>
```

Figura 1: Tela do teste

Foi constatado que somando dois valores positivos, o programa não obteve problemas.

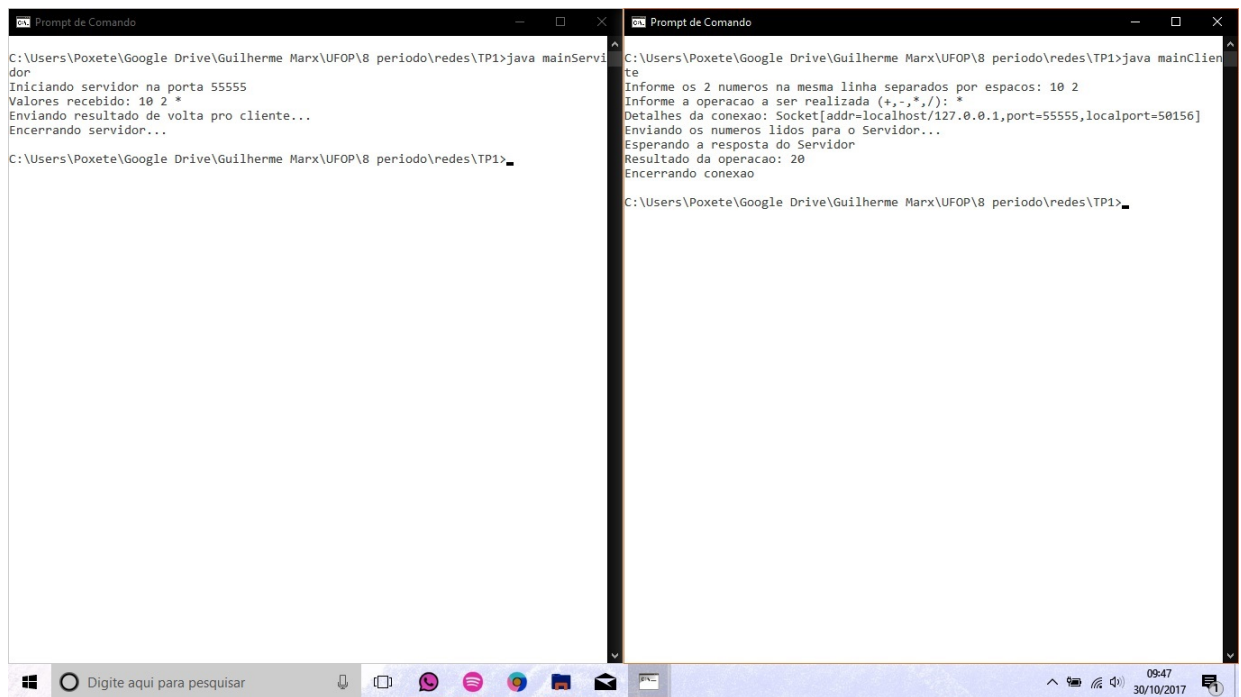
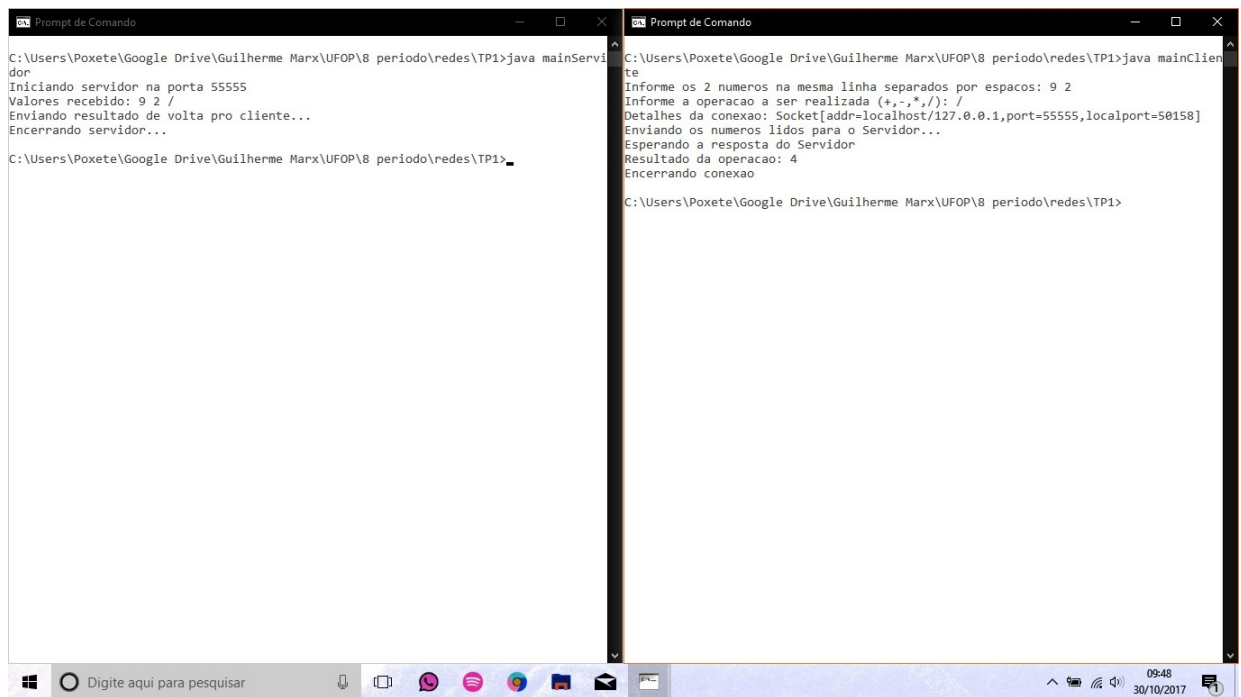


Figura 2: Tela do teste

Foi constatado que multiplicando dois valores positivos, o programa não obteve problemas.



```
C:\Users\Poxete\Google Drive\Guilherme Marx\UFOP\8 periodo\redes\TP1>java mainServidor
Iniciando servidor na porta 55555
Valores recebido: 9 2 /
Enviando resultado de volta pro cliente...
Encerrando servidor...
C:\Users\Poxete\Google Drive\Guilherme Marx\UFOP\8 periodo\redes\TP1>

C:\Users\Poxete\Google Drive\Guilherme Marx\UFOP\8 periodo\redes\TP1>java mainCliente
Informe os 2 numeros na mesma linha separados por espacos: 9 2
Informe a operacao a ser realizada (+,-,*,/): /
Detalhes da conexao: Socket[addr=localhost/127.0.0.1,port=55555,localport=50158]
Enviando os numeros lidos para o Servidor...
Esperando a resposta do Servidor
Resultado da operacao: 4
Encerrando conexao
C:\Users\Poxete\Google Drive\Guilherme Marx\UFOP\8 periodo\redes\TP1>
```

Figura 3: Tela do teste

Foi constatado que no momento da divisão, o programa desconsidera as casas decimais do resultado. Isso é pelo fato de que as variáveis dentro do programa eram inteiras, então no momento da divisão é feita uma divisão inteira e não uma divisão com parte fracionária.

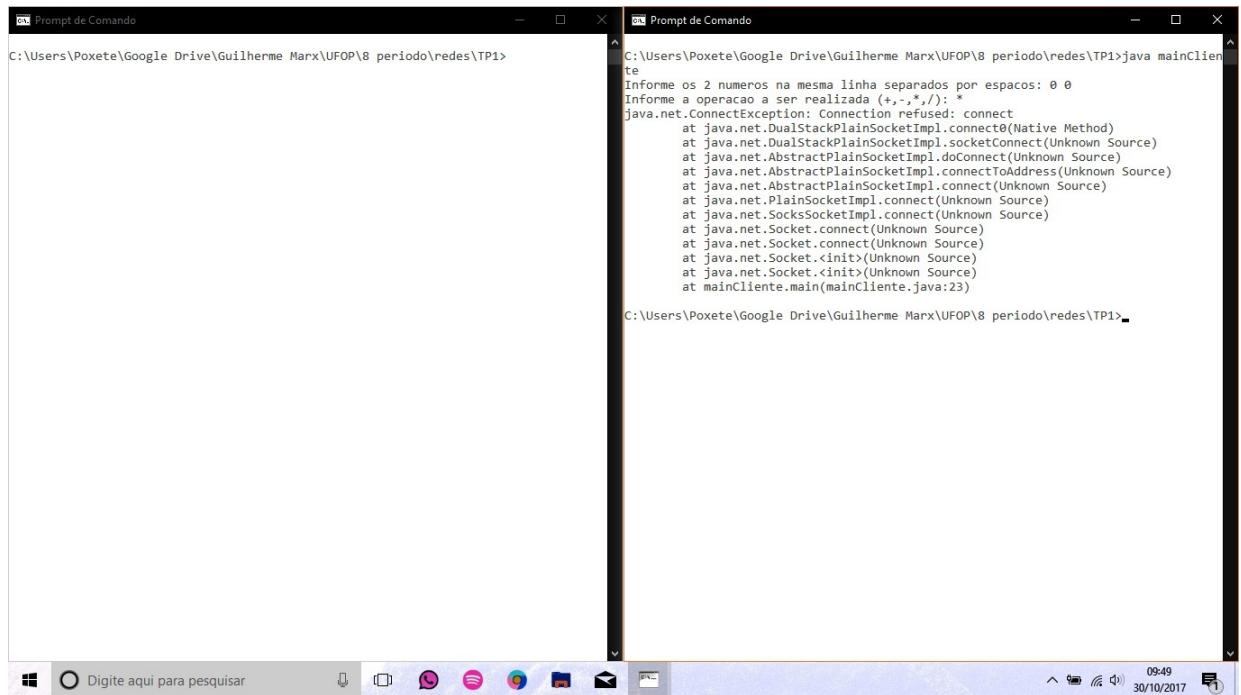


Figura 4: Tela do teste

Foi constatado que temos um crash no processo do cliente quando ele tenta conectar o servidor que não foi iniciado.

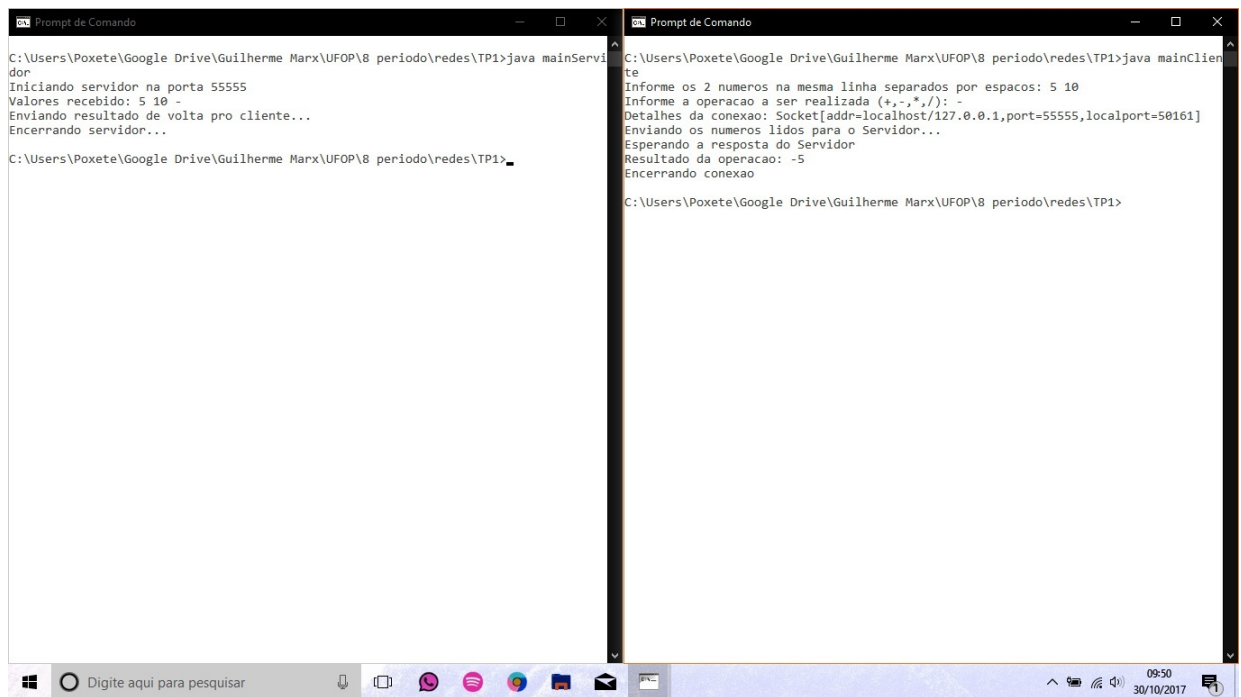


Figura 5: Tela do teste

Foi constatado que quando o resultado deve ser negativo, o programa não obteve nenhum problema.

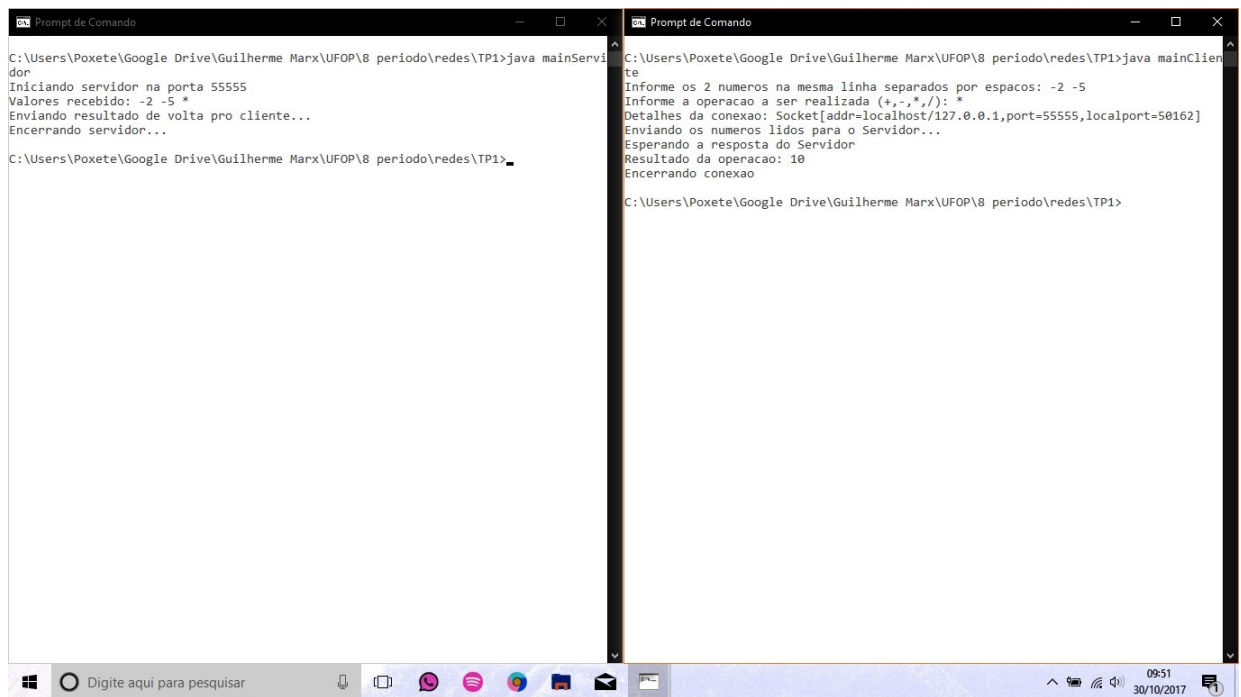


Figura 6: Tela do teste

Foi constatado que quando se usa valores negativos no programa, ele não obteve nenhum problema.

```
C:\Users\Poxete\Google Drive\Guilherme Marx\UFOP\8 periodo\redes\TP1>java mainServidor
Iniciando servidor na porta 55555
Valores recebido: 2 0 /
Exception in thread "main" java.lang.ArithmeticException: / by zero
    at mainServidor.main(mainServidor.java:33)
C:\Users\Poxete\Google Drive\Guilherme Marx\UFOP\8 periodo\redes\TP1>

C:\Users\Poxete\Google Drive\Guilherme Marx\UFOP\8 periodo\redes\TP1>java mainCliente
Informe os 2 numeros na mesma linha separados por espacos: 2 0
Informe a operacao a ser realizada (+,-,*,/): /
Detalhes da conexao: Socket[addr=localhost/127.0.0.1,port=55555,localport=50163]
Enviando os numeros lidos para o Servidor...
Esperando a resposta do Servidor
java.net.SocketException: Connection reset
    at java.net.SocketInputStream.read(Unknown Source)
    at java.net.SocketInputStream.read(Unknown Source)
    at java.net.SocketInputStream.read(Unknown Source)
    at java.io.DataInputStream.readUnsignedShort(Unknown Source)
    at java.io.DataInputStream.readUTF(Unknown Source)
    at java.io.DataInputStream.readUTF(Unknown Source)
    at mainCliente.main(mainCliente.java:36)
C:\Users\Poxete\Google Drive\Guilherme Marx\UFOP\8 periodo\redes\TP1>
```

Figura 7: Tela do teste

Foi constatado que o processo servidor dava crash quando a operação a ser realizada era uma divisão por zero. Por esse motivo, no momento da operação de divisão foi incluído um novo fluxo de execução, pois assim, consegue-se separar a divisão por zero de uma divisão normal.

```
C:\Users\Poxete\Google Drive\Guilherme Marx\UFOP\8 periodo\redes\TP1>java mainServidor
Iniciando servidor na porta 55555
Valores recebido: 10 0 /
Nao eh possivel fazer divisao por zero!!!
Enviando resultado de volta pro cliente...
Encerrando servidor...

C:\Users\Poxete\Google Drive\Guilherme Marx\UFOP\8 periodo\redes\TP1>

C:\Users\Poxete\Google Drive\Guilherme Marx\UFOP\8 periodo\redes\TP1>java mainCliente
Informe os 2 numeros na mesma linha separados por espacos: 10 0
Informe a operacao a ser realizada (+,-,*,/): /
Detalhes da conexao: Socket[addr=localhost/127.0.0.1,port=55555,localport=50165]
Enviando os numeros lidos para o Servidor...
Esperando a resposta do Servidor
Resultado da operacao: Nao foi possivel concluir a operacao... Divisao por zero encerrada
Encerrando conexao

C:\Users\Poxete\Google Drive\Guilherme Marx\UFOP\8 periodo\redes\TP1>
```

Figura 8: Tela do teste

Novo teste da divisão por zero, agora após o novo fluxo ser adicionado no servidor. No teste podemos ver que nenhum dos processos dá crash nessa situação, o problema foi resolvido.

3.1 Discussão dos resultados

Vemos com esses testes que o único problema que esse programa pode ter é quando o cliente é iniciado antes do servidor, ou seja, quando o cliente solicita a conexão ao servidor com este ainda fechado.

A divisão por zero foi relativamente simples de se resolver, colocou-se um condicional para garantir que a divisão só seja feita quando o valor do denominador não for zero e uma bandeira para avisar quando isso acontecer ou não para, caso realmente tenha-se tido uma divisão por zero, não deixar o cliente sem uma resposta.

4 Apêndice A

Programa 1: mainServidor.java

```
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;

public class mainServidor {

    public static void main(String[] args) {

        System.out.print("Iniciando servidor na porta 55555\n");
        try {

            ServerSocket servidor = new ServerSocket(55555);
            Socket novaConexao = servidor.accept(); // aceita a conexao do
                cliente

            DataOutputStream saida = new DataOutputStream(novaConexao.
                getOutputStream()); // Canal de saida
            DataInputStream entrada = new DataInputStream(novaConexao.
                getInputStream()); // Canal de entrada

            String valores = entrada.readUTF(); //recebe os valores enviados
                pelo cliente na forma "valor1 valor2 operacao"

            System.out.println("Valores recebido: " + valores);

            String[] n = valores.split(" "); //separa os valores recebidos
                cada um em uma string e a operacao em mais uma
            int valor1 = Integer.parseInt(n[0]), valor2 = Integer.parseInt(n
                [1]); //passa os valores nas strings para as variaveis
                inteiras

            int result = 0;
            boolean flag = false;
            switch(n[2]) { //le a operacao a ser realizada e a realiza
                case "+": result = valor1+valor2; break;
                case "-": result = valor1-valor2; break;
                case "*": result = valor1*valor2; break;
                case "/":
```

```

        if(valor2 == 0)
        {
            System.out.println("Nao eh possivel fazer divisao por zero!!!
                                ");
            flag = true;
            break;
        }
        result = valor1/valor2; break;
        default: break;
    }

    String resultado = new String();
    if(flag)
    {
        resultado = "Nao foi possivel concluir a operacao... Divisao
                    por zero encontrada";
    }else
        resultado = String.format("%d",result);//salva na string
                    resultado o valor calculado no switch

    System.out.println("Enviando resultado de volta pro cliente...");
    saida.writeUTF(resultado);//envia o resultado da operacao pro
        cliente

    System.out.println("Encerrando servidor...");
    novaConexao.close();//fecha a conexao com o cliente
    servidor.close();//fecha o servidor

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

}

}
}

```

Programa 2: mainCliente.java

```

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.net.Socket;
import java.util.Scanner;

public class mainCliente {

    public static void main(String[] args) {

        String numero;
        String operacao;
        boolean valoresValidos = true;
        Scanner in = new Scanner(System.in);
        System.out.printf("Informe os 2 numeros na mesma linha separados
                           por espacos: ");
        numero = in.nextLine();
    }
}

```

```

System.out.printf("Informe a operacao a ser realizada (+,-,*,/): ")
;
operacao = in.nextLine();

String dado = String.format(numero + " " + operacao);

try {
    Socket conexao = new Socket("localhost", 55555); // Conectando ao
        servidor local na porta 5555
    System.out.println("Detalhes da conexao: " + conexao.toString());

    DataOutputStream saida = new DataOutputStream(conexao.
        getOutputStream()); // Canal de saida
    DataInputStream entrada = new DataInputStream(conexao.
        getInputStream()); // Canal de entrada

    System.out.println("Enviando os numeros lidos para o Servidor..."
        );

    saida.writeUTF(dado); // Envia o numero lido para o servidor em
        formato de STRING codificada em UTF

    System.out.println("Esperando a resposta do Servidor");

    String resposta = entrada.readUTF(); // Recebe o resultado do
        Servidor em formato de STRING codificada em UTF

    System.out.println("Resultado da operacao: " + resposta);

    in.close(); // fecha o scanner
    System.out.println("Encerrando conexao");
    conexao.close(); // encerra a aplicacao do cliente
} catch (Exception e) {
    e.printStackTrace();
}
}
}

```

5 Bibliografia

- [1] TENENBAUM, AARON M. Estruturas de dados usando C / AARON M. TENENBAUM, YEDIDYAH LANHSAM, MOSHE J. AUGENSTEIN; São Paulo : MAKRON Books, 1995.
- [2] KUROSE, J. F. e ROSS, K. Redes de Computadores e a Internet / KUROSE, J. F. e ROSS, K; 5ª Ed., Pearson, 2010.
- [3] HARVEY M. DEITEL & PAUL J. DEITEL. Java como programar / HARVEY M. DEITEL & PAUL J. DEITEL, K; 8ª Ed., Prentice Hall - Br, 2010.