Nome: Bruno Felix Dias - Matrícula: 428903
Nome: Paulo Ricardo da Silva Lopes - Matrícula: 385173

# Questão 4

- Neste exercício criamos um *Chat* entre dois processos. Nós instanciamos objetos do tipo *DataOutputStream* e *DataInputStream* no processo *ChatClient* e no processo *Servidor*,

```
public class ChatClient {
        private static Scanner input = new Scanner(System.in);

        public static void main(String[] args) throws IOException {
                Socket client = null;
                DataOutputStream out;
                DataInputStream in;
                String message;
                try {
                        client = new Socket("localhost", 33000);
                        in = new DataInputStream(client.getInputStream());
                        out = new DataOutputStream(client.getOutputStream
                           ());
                        while (true) {
                                message = input.nextLine();
                                out.writeUTF(message);
                                System.out.println(in.readUTF());
                        }
                } catch (Exception e)...
```

  Então o processo *ChatClient* manda a mensagem, o processo *Servidor* mostra na tela a mensagem recebida, e enquanto isso o processo *ChatClient* fica bloqueado esperando a mensagem do processo *Servidor*.

- Neste caso o processo *Servidor* também tem a funcionalidade de enviar mensagens digitadas pelo usuário, ou seja, também funciona como um *Cliente*. E assim como no processo *ChatClient*, o processo *Servidor* também fica bloqueado aguardando a próxima mensagem ser recebida.
  Código:

```
public class Server {
        private static Scanner input = new Scanner(System.in);

        public static void main(String[] args) throws IOException {
                Socket client = null;
                DataInputStream in = null;
                DataOutputStream out = null;
                String message;
                try {
                        @SuppressWarnings("resource")
                        ServerSocket server = new ServerSocket(33000);
                        client = server.accept();
                        in = new DataInputStream(client.getInputStream());
                        out = new DataOutputStream(client.getOutputStream
                           ());
                        while (true) {
                                message = in.readUTF();
                                System.out.println(message);
```

```
18                                      message = input.nextLine();
19                                      out.writeUTF(message);
20                              }
21                      } catch (Exception e)...
```

- Conclusão: criamos um *Chat* bloqueante. Os processos sempre vão ter que aguardar um ao outro para enviar mensagens entre si.

## Referências

[GUJ ] https://www.guj.com.br/ - acessado em 20 set. 2019.

[Coulouris et al. 2013] Coulouris, G., Dollimore, J., Kindberg, T., and Blair, G. (2013). *Sistemas Distribuídos - Conceitos e Projetos*. Bookman Companhia Editora LTDA, 5ª edition.

[Tanenbaum and Steen 2008] Tanenbaum, A. S. and Steen, M. V. (2008). *Sistemas Distribuídos - Princípios e Paradigmas*. Pearson Education do Brasil, 2ª edition.