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

Questão 5

- Neste exercício nós criamos e adicionamos no *Processo Servidor* duas *Threads* que tem funções específicas. A **Send** e a *Read*.
  Código:

```
while (true) {
        client = server.accept();
        @SuppressWarnings("unused")
        Send s = new Send(client);
        @SuppressWarnings("unused")
        Read r = new Read(client);
}
```

- A *Thread Send* é responsável somente pelo envio de mensagens.
  Código:

```
public class Send extends Thread {
        Scanner input = new Scanner(System.in);
        DataOutputStream out;
        String message;

        public Send(Socket s) throws IOException {
                out = new DataOutputStream(s.getOutputStream());
                this.start();
        }
        public void run() {
                while (true) {
                        try {
                                message = input.nextLine();
                                out.writeUTF(message);
                        } catch (IOException e)...
```

- E a *Thread Read* é responsável somente pela leitura e recebimento das mensagens.
  Código:

```
public class Read extends Thread {
        DataInputStream in;
        String message;

        public Read(Socket s) throws IOException {
                in = new DataInputStream(s.getInputStream());
                this.start();
        }

        public void run() {
                while (true) {
                        try {
                                message = in.readUTF();
                                System.out.println(message);
                        } catch (IOException e)...
```

Separamos os processos e deixamos os dois executando em paralelo.

- O mesmo foi feito no processo *Cliente*, que também instancia as *Threads Send* e *Read*. Como mostrado abaixo.
  Código:

```
try {
    Socket client = new Socket("localhost", 33000);
        @SuppressWarnings("unused")
        Read r = new Read(client);
        @SuppressWarnings("unused")
        Send s = new Send(client);
} catch (Exception e)...
```

- Com isso concluímos que para deixarmos nossos processos *Não-bloqueantes* temos que usar *Threads* específicas para cada serviço.

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.