→ Sockets TCP em Java

1 – Estude e implemente a classe abaixo.

```
import java.net.*;
import java.io.*;

public class Cliente {
  public Cliente(){
  try {
    Socket sc = new Socket("127.0.0.1", 2222);

    PrintWriter pr =new PrintWriter ( sc.getOutputStream() , true);
    InputStream is = sc.getInputStream();
    BufferedReader br = new BufferedReader (new InputStreamReader(is));

    System.out.println(br.readLine());
    pr.println(" Olá, eu sou o cliente");

    System.out.println(br.readLine());
    pr.println("Cliente: Continuo por aqui");

    pr.close();
    is.close();
    sc.close();
  }
  catch (IOException e){
   System.out.println( e.getMessage());
  }
  }
  public static void main (String args[]){
    Cliente c=new Cliente();
  }
}
```

a) O que acontece se tentar executar este programa?

b) Implemente uma classe, Servidor, que comunique com o Cliente do exercício anterior.

(Nota: Consulte os apontamentos da aula teórica, T02, p.31 e seguintes)

c) Teste a aplicação cliente-servidor que construiu.

d) – Depois de executar o cliente e o servidor anteriores na mesma máquina, teste o seu processo Cliente com o processo Servidor de um dos seus colegas e vice-versa.

e) Modifique o Cliente e o Servidor das alíneas a) e b) de forma a ambos efectuarem primeiro as operações de leitura no socket e depois as operações de escrita. O que acontece?

f) Modifique agora o Cliente e o Servidor dos exercícios 1 e 2 de forma a ambos efectuarem primeiro as operações de escrita no socket e depois as operações de leitura. O que acontece?

2 – Modifique o exercício anterior (alíneas a e b) de forma a usar Streams de **objectos** em vez de Streams de caracteres.

3 – Modifique o Servidor do exercício **2** de maneira a que este permaneça em execução à espera da ligação de novos clientes, isto é, depois de conversar com um cliente, aceita a ligação de um outro cliente. Teste o programa com um colega.

4 – Pretende-se construir uma aplicação cliente/servidor cuja comunicação é feita através de Sockets. O processo Servidor recebe um valor do tipo char que identifica o tipo de cliente que está a comunicar com ele.
Clientes do tipo 'A', depois de enviarem ao servidor o seu tipo, enviam um valor inteiro e recebem como resposta o quadrado desse valor.
Clientes do tipo 'B', depois de enviarem ao servidor o seu tipo, enviam um valor do tipo double e recebem como resposta a raiz quadrada desse valor.

a) Construa as classes Servidor, Cliente_A e Cliente_B usando **ObjectStreams**. Os clientes devem poder fazer vários pedidos ao servidor até decidirem terminar. Escolha um critério de paragem.

b) Modifique o servidor para que o servidor do exercício anterior aceite vários clientes, uns após os outros.

5 – Construa uma aplicação cliente – servidor, com comunicação por Sockets TCP, que permita dois utilizadores manterem uma conversa em que as mensagens devem ser linhas de texto (Strings) introduzidas através dos respetivos teclados. Na comunicação use ObjectStreams.
A conversa deverá seguir o seguinte protocolo:
. O utilizador do processo cliente é o primeiro a "falar";
. O utilizador do processo servidor decide quando termina a conversa enviando uma linha de texto com a palavra **fim**.
- Quando o utilizador do processo servidor termina a conversa, o cliente termina, mas o processo servidor deverá continuar a execução esperando que um novo cliente estabeleça ligação.

- Teste o programa com duas máquinas.

- **Implementação da invocação de métodos de um objeto remoto através de sockets**

6 - Construa uma classe Aluno em que cada aluno tem um número (correspondente ao seu número de aluno) um nome, o curso que frequenta e um contacto (telefone ou e-mail).
- Defina as operações básicas de consulta e modificação (getters e setters) para cada um dos atributos e os métodos toString e equals (*pode usar a opção "insert code" do NetBeans*).

- Pretende-se uma aplicação, cliente - servidor, em que o processo servidor faz a gestão de uma lista de alunos. Essa lista (objecto do tipo ArrayList<Aluno>) deve ser mantida em memória e, sempre que é modificada, ser escrita para um ficheiro.)
O processo cliente permitirá o seguinte conjunto de operações:

i) Registar aluno (operação em que envia os dados de um aluno ao servidor e recebe como resposta o número total de alunos registados até ao momento). Deverá ser verificado se o aluno ainda não está registado.

ii) Consultar quais os alunos registados (Obter uma lista com os dados de todos os alunos já registados.

iii) Consultar o número de acessos ao servidor ocorridos até ao momento.

iv) Dado um nome de aluno, devolver o seu número e o seu contacto, ou caso haja mais de um aluno com o mesmo nome devolver os vários números e contactos.

A classe que implementa o processo Servidor possui os atributos:

- alunosRegistados, que será um objecto do tipo java.util.ArrayList<Aluno>, e irá conter os dados de todos os alunos registados;

- numeroAcessos – valor inteiro que representa o número de acessos ao servidor;

O servidor contém também os métodos que implementam cada uma das operações a que o processo cliente pode aceder.

a) Implemente a aplicação usando Sockets TCP, e as Streams (ObjectInputStream e ObjectOutputStream).
- O processo cliente deve poder executar várias opções até decidir terminar; Adicione uma opção de "Sair"
- o processo servidor, após responder aos pedidos de um cliente, deve poder atender outro cliente.

b)Teste com vários clientes a aceder em simultâneo ao servidor. O que acontece?