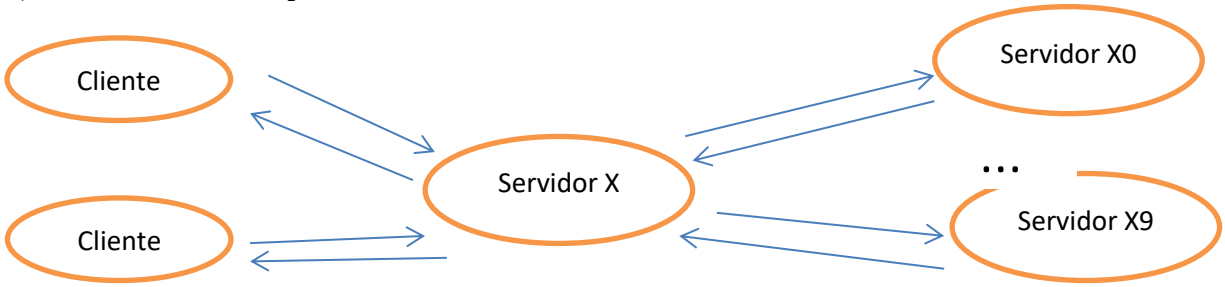
Nota: Nos exercícios 1 e 2, pode omitir o tratamento de exceções.

1 – Suponha uma aplicação cliente/servidor em que o servidor possui um objeto remoto que faz a gestão de uma lista de nomes. As operações do objeto remoto devem ser as seguintes:
1) Inserir um nome; os nomes devem ser guardados ordenados alfabeticamente por ordem crescente, e quando o nome já existir na lista não deve ser inserido. No caso de a inserção ter sucesso o método deve devolver o valor "true", caso contrário devolve "false".
2) Apagar um nome; se a operação for realizada com sucesso o método deve devolver o valor "true", caso contrário devolve "false".
3) Consultar lista de nomes.
4) Consultar número total de acessos, isto é, consultar o número total de invocações remotas no objeto.
a) Construa a interface remota do objeto
b) Construa a classe que implementa o objeto remoto.
c) Construa a classe que implementa o processo Servidor que conterá o objeto remoto.
d) Construa uma classe Cliente que permita testar os métodos remotos. A classe não tem de apresentar opções ao utilizador. Apenas terá de invocar os 3 métodos remotos definidos e mostrar o seu resultado.

2 - Suponha uma aplicação distribuída com um processo que funciona como distribuidor de invocações, (Servidor X, da figura). Para cada cliente que acede ao Servidor X, é criada uma Thread para servir o pedido desse cliente. Cada pedido de um cliente consiste numa String sem significado e irá ter como resposta o conteúdo de um ficheiro gerido por um servidor auxiliar (Servidor Xi).
O Servidor X ao receber um pedido de um cliente irá aceder a um servidor auxiliar (Servidor Xi) que irá ler o conteúdo de um ficheiro de nome "DADOS.dat". O servidor auxiliar enviará para o servidor X o conteúdo do ficheiro e por sua vez o servidor X enviará o conteúdo do mesmo ficheiro ao cliente que lhe fez o pedido.
O servidor X contém uma tabela com 10 endereços (IP's), cada um correspondendo a um servidor auxiliar. Os pedidos dos clientes irão sendo direccionados sucessivamente para o servidor auxiliar 0, 1, 2 .. 9. Ao chegar ao 11º cliente, volta ao servidor auxiliar 0, e assim por diante. Supondo que a comunicação cliente / servidor x é feita através de sockets e que a comunicação servidor x / servidor xi é feita por RMI, construa:
a) A classe cliente
b) A classe Servidor X
c) A interface remota para o servidor Xi, e a classe Servidor Xi.



Notas auxiliares:

socket do cliente:

import java.net.*;

import java.io.*;

Socket meuCliente = null;

try { meuCliente = new Socket ("host", portNumber); }

catch (IOException e){ … }

Obter as Streams do socket e associar Object Strems:

```
ObjectOutputStream os = new ObjectOutputStream (
                                        meuCliente.getOutputStream());

ObjectInputStream is = new ObjectInputStream (
                                        meuCliente.getInputStream());
```

String s = "exemplo";

os.writeObject(s);

s = (String) is.readObject();

socket do servidor:

ServerSocket meuServidor = null;

try { meuServidor = new ServerSocket (portNumber); }

catch (IOException e){ …}

Socket sServidor = null

try { sServidor = meuServidor.accept(); }

catch (IOException e){ }

RMI:

Interface remota: java.rmi.Remote

Exceção remota: java.rmi.RemoteException

Objeto remoto : java.rmi.server.UnicastRemoteObject;

Sintaxe do nome que o objecto remoto tem no RMIregistry:

[rmi:] [//] [nomeMaquina] [:port] [/nomeObjecto]

Instalar um gestor de segurança:

System.setSecurityManager (new SecurityManager());

Iniciar o registry: java.rmi.registry.LocateRegistry.createRegistry(1099);

Métodos da classe java.rmi.Naming:

void rebind (String nomeObjecto, Remote objecto);

Remote lookup (String nomeObjecto)

Geração de valores aleatórios:

Gerador de números pseudo-aleatórios da classe Math: Math.random().

Este método devolve um valor do tipo double pertencente ao intervalo [0, 1[

Class java.util.ArrayList:

```
ArrayList() // construtor vazio, dimensão inicial zero.
boolean add(Object element)
// adiciona o elemento especificado ao final da lista
void add( int index, Object obj)
//insere o elemento especificado na posição index
Object remove(int index )//remove o elemento da posiçã index
boolean  remove( Object o)
//remove a primeira ocorrência do objecto dado como parâmetro
Object set (int position, Object obj )
//  substitui o elemento da posição index pelo elemento dado
Object get (int position)//devolve o elemento da posição index

void clear() // remove todos os elementos da lista
Object clone() // devolve uma cópia da lista
boolean contains(Object element)
// devolve true se a lista contém o elemento especificado
boolean equals ( Object obj)
// permite comparar duas listas
int indexOf(Object element)
 // procura o índice da 1ª ocorrência de elemento
boolean isEmpty() // verifica se a lista não tem componentes
int size() // devolve a dimensão actual
String toString ()
```