# *Exame da Época Especial – Avaliação prática – P1*

Pretende-se que desenvolva, em linguagem Java, um método atendendo aos seguintes requisitos:

- Protótipo do método pretendido:

  ***void processRequest(Socket s) throws Exception***;

- O *socket* TCP ***s*** já se encontra criado e conectado a um par remoto, estando pronto para o envio e recepção de dados;

- O médoto ***processRequest*** vai recebendo objectos serializados do tipo ***Request*** através do *socket* ***s*** até que ocorra uma excepção qualquer;

- A classe ***Request*** possui, entre outros, os métodos ***int getUdpPort()***, ***String getIpAddress()*** e ***String getMsg()***;

- Para cada objecto ***Request*** recebido, a *string* devolvida pelo método ***getMsg()*** é enviada, através do protocolo UDP (***DatagramSocket***) e em formato texto (i.e., sequência de caracteres) para o destino com o endereço IP e o porto estipulados;

- Quando ocorre uma excepção de qualquer tipo, o método ***processRequest***:

  - Encerra o *socket* ***s***;
  - Apresenta a mensagem associada à excepção;
  - Volta a lançar a excepção, o que faz com que termine.

***Notas:***

- ***Na sua resposta, mão é necessário indicar os imports / packages das classes usadas;***
- ***Deve indentar correctamente o código.***

```
import java.net.*;

void processRequest(Socket s) throws Exception

{

      . . .

      while(true)

      {

            /* Receive a serialized Request object */

            . . .

            /* Transmit the specified message to the specified IP
               address and UDP port.

               The message is transmitted as a sequence of
               characters (i.e., it is not transmitted as a
               serialized String object).

            */

            . . .

      }

}
```

# *Exame da Época Especial – Avaliação prática – P2*

Pretende-se que desenvolva um serviço remoto Java RMI, com um mecanismo de ***callback***, atendendo aos seguintes requisitos:

- Nome da classe que representa o serviço: ***MessageReflector***;
- Interface remota implementada pelo serviço *MessageReflector*:
  - Nome: ***MRInterface***;
  - Métodos:
    - ***boolean registerClient (MRClientInterface cliRef)***;

      Se ainda não existir, acrescenta a referência RMI passada como argumento a uma lista interna e devolve ***true***. Se já existir, ignora o pedido e devolve ***false***.

      *Notas:*

      1. *Em vez de uma lista, também pode recorrer a um conjunto;*
      2. *Cada cliente implementa um serviço RMI baseado na interface remota MRClientInterface.*

    - ***boolean unregisterClient (MRClientInterface cliRef)***;

      Se existir, remove a referência RMI passada como argumento da lista interna e devolve ***true***. Caso contrário, devolve ***false***.

    - **void broadcastMessage(String msg)**;

      Comunica a *string* passada como argumento a todos os clientes registados. Para o efeito, invoca o método ***void postMessage(String msg)*** pertencente à interface remota ***MRClientInterface***. Qualquer problema que surja na invocação do método *postMessage* leva à eliminação da respectiva referência remota (i.e., do cliente) da lista interna.

    - ***int getNumRegisteredClients()***;

      Devolve o número de clientes registados (i.e., o número de referências remotas existentes na lista interna).

***Notas:***

- ***Na sua resposta, mão é necessário indicar os imports / packages das classes usadas;***
- ***Não deve implementar a interface remota "MRClientInterface" nem qualquer serviço RMI associado a esta ou a qualquer cliente;***
- ***Não deve implementar qualquer método "main" nem criar ou registar qualquer serviço "MessageReflector" num servidor "RMI registry";***
- ***Deve indentar correctamente o código.***