→ A classe InetAddress

1 – O programa abaixo permite-lhe obter o nome da máquina onde está a trabalhar. Implemente-o e explore a classe InetAddress.

```
import java.net.*;
public class GetName {
  public static void main (String args[]) throws Exception{
    InetAddress host = null;

    host = InetAddress.getLocalHost();
    System.out.println(host.getHostName());
}   }
```

2 – O programa abaixo permite-lhe obter o endereço IP da máquina onde está a trabalhar. Implemente-o e estude o seu comportamento.

```
import java.net.*;
public class GetIP {
  public static void main (String args[]) throws Exception{
    InetAddress host = null;
    host = InetAddress.getLocalHost();
    byte ip [] = host.getAddress();
    for ( int i= 0 ; i < ip.length; i++){
      if (i>0) System.out.print(".");
      System.out.print(ip[i] & 0xff);
    }
    System.out.println();
  }
}
```

3 – Na classe InetAddress explore os métodos getByName, getHostName e getHostAddress. Tente **completar** os exercícios seguintes.

a) Pretende-se que o programa abaixo, dado um determinado IP, nos diga qual o nome da máquina correspondente.

```
import java.net.*;
import java.io.*;
public class IPtoName {
 public static void main (String args[] ) throws IOException{
            String s=" ";
            char c;
            System.out.print("IP address? ");
            while ( (c=(char)System.in.read()) != 10)
                        s+=c;
            s=s.trim();
       InetAddress host =null;
       try {
```

```
        < exercício >
       }
       catch (UnknownHostException e){
         System.out.println("IP malformed ");
       }
     }
}
```

b) Construa um programa que dado o nome de uma máquina, nos diga qual o IP correspondente.

c) No programa da alínea b) experimente dar como input a string "google.com". O que acontece?

→ Sockets UDP em Java

4 – As duas classes que se seguem implementam dois processos que comunicam por sockets UDP (*Transmission Control Protocol)*

Um Socket é uma interface de um canal de comunicação entre dois processos. Um par de processos comunica através de um par de sockets ver figura 1.

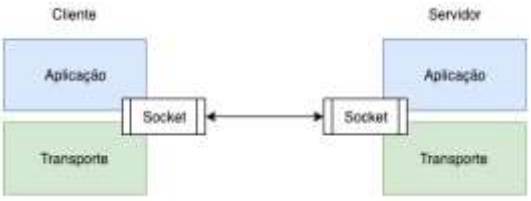

Figura 1 – Comunicação por sockets. Fonte: [1].

O endereço de um socket é especificado por:
- endereço internet, isto é, o IP da máquina onde está o processo com que queremos comunicar;
- número de um porto, onde um porto é representado por um inteiro >1024 que identifica os vários serviços em execução numa mesma máquina, os valores de 0 a 1023 são reservados a utilizadores com privilégios root ou superuser.

Ex. lo 146.75.4.30/1234

O protocolo de comunicação UPD (ver figura 2), como deve ter estudado em Redes de Computadores é um protocolo de transporte, sem conexão, que não garante a entrega de todos os pacotes de dados nem garante a entrega por ordem de envio.

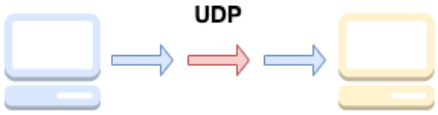

Figura 2 – Comunicação por UDP. Fonte: [1].

- A classe DatagramSocket permite implementar um socket UDP em java e a classe DatagramPacket permite criar um pacote de dados a enviar por UDP.

a) Estude e implemente num mesmo projecto, as duas classes abaixo, UDPClient e UDPServer. Repare que agora tem dois programas, isto é, duas funções *main*. Quando executar a sua aplicação, vai ter dois processos em execução e estes vão comunicar entre si.

- Para executar a aplicação, comece por colocar em execução o processo UDPServer. Depois inicie a execução do UDPClient. À pergunta "Qual o servidor?" pode introduzir o IP da sua máquina ou o IP 127.0.0.1, que permite comunicar com a própria máquina.

```
import java.net.*;
import java.io.*;
public class UDPClient{
   public static String readString (){
     BufferedReader canal;
     canal = new BufferedReader ( new InputStreamReader (System.in));
     try {
        return canal.readLine();
     }
     catch (IOException ex) {
        return null;
     }
  }
  public static void main(String args[]){
        String s;
        System.out.print("Qual o servidor? ");
        String host = readString();
        DatagramSocket aSocket = null;
          try {
               aSocket = new DatagramSocket();
                while(true){
                   System.out.print("<Client>  Mensagem a enviar = ");
                   s = readString();
                   byte [] m = s.getBytes();
                   InetAddress aHost = InetAddress.getByName(host);
                   int serverPort = 2222;
                   DatagramPacket request = new DatagramPacket(m, m.length, aHost,serverPort);
                   aSocket.send(request);
                   byte[] buffer = new byte[100];
                   DatagramPacket reply = new DatagramPacket(buffer, buffer.length);
                   aSocket.receive(reply);
                   System.out.println("<Client>  Recebeu: " + new String(reply.getData()));
               } // while
          }catch (SocketException e){System.out.println("Socket: " + e.getMessage());
          }catch (IOException e){System.out.println("IO: " + e.getMessage());
        }finally {if(aSocket != null) aSocket.close();}
   }
}
```

```
import java.net.*;
import java.io.*;
public class UDPServer{
      public static void main(String args[]){
      DatagramSocket aSocket = null;
      String s;
         try{
            aSocket = new DatagramSocket(2222);
            System.out.println("<Server>  Socket Datagram à escuta no porto 2222");
            while(true){
               byte[] buffer = new byte[100];
               DatagramPacket request = new DatagramPacket(buffer, buffer.length);
               aSocket.receive(request);
               s=new String(request.getData());
               System.out.println("<Server>  Server Recebeu: " + s);
               DatagramPacket reply = new DatagramPacket(request.getData(),
                        request.getLength(), request.getAddress(), request.getPort());
               aSocket.send(reply);
            }
          }catch (SocketException e){System.out.println("Socket: " + e.getMessage());
         }catch (IOException e) {System.out.println("IO: " + e.getMessage());
      }finally {if(aSocket != null) aSocket.close();}
   }
}
```

a) Experimente eliminar (terminar) o processo cliente, o que acontece?

b) E se eliminar (terminar) o processo servidor?

c) Investigue quais as alterações que deve colocar no cliente para este gerar um timeout quando não recebe uma resposta do servidor no espaço de 10 segundos?

d) Modifique a classe cliente para terminar quando o utilizador escrever a palavra “fim”.

e) Altere a classe cliente de forma a calcular o RTT (Round-Trip-Time) da comunicação, isto é, o tempo total que uma mensagem demora a ir de um processo, A, a um processo B e regressar a A.

Nota: Para medir o tempo pode usar o seguinte método:
long t=System.currentTimeMillis(); // Devolve o tempo actual em mili-segundos.

- Se estiver a testar em apenas uma máquina o RTT vai muito próximo de 0. Pode colocar o servidor a fazer algum trabalho antes de responder ao cliente.

f) Modifique o programa anterior para que a String a enviar pelo servidor seja lida do teclado. Para ler a String pode usar a função umaString() da classe Ler que criou em POO. Ver página 4 de https://www.di.ubi.pt/~pprata/poo/POO_22_23_FP02.pdf

g) Estude, nessa ficha, FP02, como usar a classe Ler noutros projetos do IDE eclipse.

[1] https://www.treinaweb.com.br/blog/uma-introducao-a-tcp-udp-e-sockets/