EJERCICIO 1

Realiza todos los ejemplos de los apuntes y comprueba que funcionan correctamente entendiendo todo lo que hace cada uno de los ejemplos.

EJERCICIO 2

Basándote en la información de los apuntes y en la búsqueda de información en Internet. Responde lo mejor que puedas a las siguientes preguntas.

- ¿qué propósito tiene dividir las comunicaciones por red en capas?
- ¿qué motiva la aparición de un protocolo como DNS?
- ¿cuál es la utilidad del concepto de "puerto"?
- ¿cuál es la utilidad del concepto de "socket"?
- ¿qué similitudes y diferencias existen entre un proxy y un cortafuegos?
- ¿qué relación existe entre la idea de "servicio" y "puerto"?
- ¿qué similitudes y diferencias existen entre los protocolos de capa de transporte TCP y UDP?
- ¿qué similitudes y diferencias funcionales existen entre los protocolos HTTP y FTP?

EJERCICIO 3

El siguiente código java inicia un servidor en el puerto 9999. Estudia qué es lo que hace este servidor. Impleméntalo en el IDE empleado en clase y después crea una clase Principal que en su método main ejecute al método inicia de una instancia de la clase Conector, de modo que deje al servidor preparado para recibir conexiones de clientes.

Conector.java

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

public class Conector {
   ServerSocket server;
   Socket socket;
   int puerto = 9999;
   DataOutputStream salida;
   BufferedReader entrada;

   public void inicia(){
    try{
       server = new ServerSocket(puerto);
       socket = new Socket();
       socket = server.accept();

       entrada = new BufferedReader(new InputStreamReader(socket.getInputStream()));
       String mensaje =  entrada.readLine();
       System.out.println(mensaje);
       salida = new DataOutputStream(socket.getOutputStream());
       salida.writeUTF("Adios");
       socket.close();
    }catch (IOException e){
       System.out.println(e.getMessage());
    }catch(Exception e){
       System.out.println(e.getMessage());
    }
   }
}
```

Una vez ejecutado el servidor y dejado preparado, abre un terminal de comandos y ejecuta la siguiente sentencia:

- `telnet 127.0.0.1 9999`

con esto nos estamos conectando al servidor creado anteriormente y se abrirá un nuevo terminal, “Telnet 127.0.0.1”, vacío que será el cliente que está accediendo a ese servidor, en él podemos escribir el mensaje que se desea enviar al servidor.

En caso de que no funcione el comando telnet realiza lo siguiente:

> Microsoft Windows 7 no instala el cliente telnet por defecto, si lo necesitas tienes que instalarlo manualmente. Te explicamos cómo hacerlo:
>
> 1. Accede a "Iniciar" - "Panel de control" - "Programas y características".
> 2. En la ventana de agregar o quitar software pulsa en "Activar o desactivar las características de Windows".
> 3. Marca la característica "Cliente Telnet" y pulsa "Aceptar".

El siguiente código es un cliente que se conecta al servidor anterior.

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

public class Cliente {
   Socket cliente;
   int puerto = 9999;
   String ip = "127.0.0.1";
   BufferedReader entrada, teclado;
   PrintStream salida;

   public void inicia(){
    try{
        cliente = new Socket(ip,puerto);
        teclado = new BufferedReader(new InputStreamReader(System.in));
        entrada = new BufferedReader(new InputStreamReader(cliente.getInputStream()));
        System.out.println("Escribe el mensaje para el servidor");
        String tec = teclado.readLine(); //texto de entrada por teclado

        salida = new PrintStream(cliente.getOutputStream()); //canal de salida
        salida.println(tec);//enviamos el mensaje que se ha introducido por teclado

        String mensaje =  entrada.readLine();//se espera a recibir respuesta
        System.out.println(mensaje);//se muestra la respuesta recibida

        //se cierran todos los flujos, canales y conexiones
        entrada.close();
        salida.close();
        teclado.close();
        cliente.close();
    }catch (IOException e){
        System.out.println(e.getMessage());
    }catch(Exception e){
        System.out.println(e.getMessage());
    }
   }
}
```

Para ejecutar este código debes ejecutar mediante terminal (cmd) al menos uno de los dos, el cliente o el servidor.
Ejecuta primero el servidor y a continuación el cliente, enviando el mensaje que quieras al servidor, comprueba que todo funciona correctamente.

Realiza ahora lo siguiente:

A. Amplia el código de la parte servidor haciendo uso tanto de la clase “Socket” como de la clase InetAddress con objeto de obtener por pantalla dos tipos de información:

a) consola del cliente: el programa desde la parte cliente ofrecerá la posibilidad de introducir mediante teclado una URL al usuario con su nombre DNS. El programa traducirá el nombre y devolverá su IP asociada al cliente.

b) consola del servidor: se deberá mostrar por pantalla (del cliente), deberá ofrecer además de la IP equivalente al nombre DNS, una frase del tipo ...

```
Conectado a /x.y.z.w en el puerto 9999 desde el puerto ‘b’ de /x.y.z.w
```

… donde la primera dirección es la dirección del servidor, siendo ‘9999’ su puerto de conexión, ‘b’ es el puerto del cliente y la segunda dirección es la del equipo del cliente también.

B. Se pide modificar el código de la parte de servidor de forma que la funcionalidad del método “inicia” se distribuya en los siguientes métodos:

- iniciar: creará el socket y abrirá la conexión.
- informar: gestionará el intercambio de información entre cliente y servidor, mostrando por pantalla los datos sugeridos anteriormente.
- terminar: cerrará las conexiones y “streams” abiertos.