UD 3 – PROGRAMACIÓN DE COMUNICACIONES EN RED

BOLETÍN

INETADDRESS

1. Crear una aplicación Java que muestre la dirección IP actual de la máquina.
2. Crear una aplicación Java que muestre la dirección IP de una máquina de la red introduciendo su nombre como parámetro al iniciar la aplicación.
3. Crear una aplicación que nos muestre la dirección de loopback de la máquina.

4. Crear una aplicación Java que introduciendo como parámetro una dirección IP nos diga si es una dirección multicast o no.

5. Crear una aplicación Java que introduciendo como parámetro una dirección web, nos muestre la dirección ip del servidor en que está alojada.

6. Crear una aplicación Java que aúna todas las funciones previamente desarrolladas mediante un menú de usuario que permite a éste elegir qué acción realizar. El menú contextual será el siguiente:
    a. Visualizar las direcciones IP del dispositivo.
    b. Visualizar las direcciones IP de un dispositivo indicado por su nombre.
    c. Visualizar la dirección de *loopback* del dispositivo.
    d. Visualizar solo la dirección IPv4 del dispositivo.
    e. Visualizar solo la dirección iPv6 del dispositivo.
    f. Visualizar el nombre del dispositivo y la dirección IP.
    g. Visualizar el nombre canónico completo de la máquina.

7. Crear una aplicación multihilo, que:
    a. Una vez ejecutada monitoriza cada 4 segundos las direcciones IP del dispositivo.
    b. Las almacena en un fichero de texto junto con el *timestamp* del momento en que se han obtenido.
    c. Cada iteración se almacenará en una línea diferente.
    d. El programa se cerrará cuando el usuario pulse la tecla ENTER.

8. Crear una aplicación que permita hacer un “ping” a una dirección IP dada por el usuario. No será válido crear un proceso para que lo haga el comando “ping”.

SOCKETS TCP

9. Crear una aplicación Java cliente que se conecte al NIST (*time.nist.gov* puerto *13*) para obtener la fecha y hora actual.

10. Crear un servicio en red mediante una aplicación Java cliente y una aplicación Java servidor que tenga la siguiente funcionalidad:
    a. **Servidor**:
        i. Muestra los siguientes datos del servidor:
            1. Nombre.
            2. Dirección IP.
            3. Puerto de escucha.
        ii. Muestra por pantalla la dirección IP y el nombre del cliente, así como el puerto en el que está conectado en el servidor.
        iii. Muestra por pantalla el mensaje de texto del cliente.
        iv. Devuelve al cliente el mensaje "La hora actual es: " junto con la hora actual dentro del mensaje de texto.
    b. **Cliente**:
        i. Muestra los siguientes datos del servidor:
            1. Nombre.
            2. Dirección IP.
            3. Puerto del socket propio.
        ii. Envía al servidor el mensaje de texto "Cliente intentando adquirir la hora actual.".
        iii. Muestra por pantalla el mensaje "Intentando conectar con el servidor: *nombre* con dirección IP: *dirección* para obtener la hora."
        iv. Muestra por pantalla el mensaje devuelto por el servidor y termina.

11. Crear un servicio en red mediante una aplicación Java cliente y una aplicación Java servidor que tenga la siguiente funcionalidad:
    a. **Servidor**:
        i. Muestra los siguientes datos del servidor:
            1. Nombre.
            2. Dirección IP.
            3. Puerto del socket propio.
        ii. Muestra por pantalla la dirección IP y el nombre del cliente, así como el puerto en el que está conectado.
        iii. Muestra por pantalla el mensaje de texto del cliente.

iv. Devuelve al cliente un mensaje con el valor de un número entero aleatorio entre 1 y 1000 obtenido de ejecutar una aplicación separada con dicha funcionalidad. (Para esto deberás recordar los conceptos del primer tema "*Programación multiproceso*").

b. **Cliente**:
   i. Muestra los siguientes datos del servidor:
      1. Nombre.
      2. Dirección IP.
      3. Puerto del socket propio.
   ii. Envía al servidor el mensaje de texto "Cliente intentando adquirir un número aleatorio.".
   iii. Muestra por pantalla el mensaje "Intentando conectar con el servidor: *nombre* con dirección IP: *dirección* para obtener un número aleatorio entre 1 y 1000. "
   iv. Muestra por pantalla el mensaje devuelto por el servidor y termina.

12. Crear un servicio en red mediante una aplicación Java cliente y una aplicación Java servidor que tenga la siguiente funcionalidad:
    a. **Servidor**:
       i. Muestra los siguientes datos del servidor:
          1. Nombre.
          2. Dirección IP.
          3. Puerto del socket propio.
       ii. Muestra por pantalla la dirección IP y el nombre del cliente, así como el puerto en el que está conectado.
       iii. Muestra por pantalla el mensaje de texto del cliente.
       iv. **Analiza el mensaje del cliente**:
          1. Si el mensaje del cliente es distinto de "FIN", devuelve al cliente un mensaje con el valor de un número entero aleatorio entre 1 y 1000 obtenido de ejecutar una aplicación separada con dicha funcionalidad. (Para esto deberás recordar los conceptos del primer tema "*Programación multiproceso*").
          2. Si el mensaje del cliente es "FIN", termina su ejecución.
    b. **Cliente**:
       i. Muestra los siguientes datos del servidor:
          1. Nombre.
          2. Dirección IP.
          3. Puerto del socket propio.

ii. Pide al usuario un valor entero entre 1 y 1000.
iii. **Función cíclica** que termina cuando el servidor devuelve un número igual a un número previo elegido por el usuario.
    1. Envía al servidor el mensaje de texto “Cliente intentando adquirir un número aleatorio.”.
    2. Muestra por pantalla el mensaje “Intentando conectar con el servidor: *nombre* con dirección IP: *dirección* para obtener un número aleatorio entre 1 y 1000. ”
    3. Muestra por pantalla el mensaje devuelto por el servidor.
    4. Analiza el valor devuelto:
        a. Si es el mismo valor que el seleccionado por el usuario:
            i. El programa muestra por pantalla el mensaje “¡Valor acertado por el servidor!, Terminando programa”.
            ii. Envía al servidor el mensaje “FIN” y termina.
        b. Si no es el valor seleccionado por el usuario:
            i. Se repite el proceso.

13. Repetir el problema “14” pero con las modificaciones necesarias para que después de terminar de atender a un cliente, siga disponible para atender a otros clientes.
    a. El servidor **se cerrará** o bien después de atender a 3 clientes o bien cuando el usuario administrador escriba por pantalla “FIN”.

SOCKETS UDP

14. Crear un servicio en red mediante una aplicación Java cliente y una aplicación Java servidor que tenga la siguiente funcionalidad:
    a. **Servidor**:
        i. Muestra los siguientes datos del servidor:
            1. Nombre.
            2. Dirección IP.
            3. Puerto de escucha.
        ii. Muestra por pantalla la dirección IP y el nombre del cliente, así como el puerto en el que está conectado en el servidor.
        iii. Muestra por pantalla el mensaje de texto del cliente.
        iv. Devuelve al cliente el mensaje "La hora actual es: " junto con la hora actual dentro del mensaje de texto.
    b. **Cliente**:
        i. Muestra los siguientes datos del servidor:
            1. Nombre.
            2. Dirección IP.
            3. Puerto del socket propio.
        ii. Envía al servidor el mensaje de texto "Cliente intentando adquirir la hora actual.".
        iii. Muestra por pantalla el mensaje "Intentando conectar con el servidor: *nombre* con dirección IP: *dirección* para obtener la hora."
        iv. Muestra por pantalla el mensaje devuelto por el servidor y termina.

15. Crear un servicio en red mediante una aplicación Java cliente y una aplicación Java servidor que tenga la siguiente funcionalidad:
    a. **Servidor**:
        i. Muestra los siguientes datos del servidor:
            1. Nombre.
            2. Dirección IP.
            3. Puerto del socket propio.
        ii. Muestra por pantalla la dirección IP y el nombre del cliente, así como el puerto en el que está conectado.
        iii. Muestra por pantalla el mensaje de texto del cliente.
        iv. **Analiza el mensaje del cliente**:
            1. Si el mensaje del cliente es distinto de "FIN", devuelve al cliente un mensaje con el valor de un número entero

aleatorio entre 1 y 1000 obtenido de ejecutar una aplicación separada con dicha funcionalidad. (Para esto deberás recordar los conceptos del primer tema "*Programación multiproceso*").

2. Si el mensaje del cliente es "FIN", termina su ejecución.

b. **Cliente**:
   i. Muestra los siguientes datos del servidor:
      1. Nombre.
      2. Dirección IP.
      3. Puerto del socket propio.
   ii. Pide al usuario un valor entero entre 1 y 1000.
   iii. **Función cíclica** que termina cuando el servidor devuelve un número igual a un número previo elegido por el usuario.
      1. Envía al servidor el mensaje de texto "Cliente intentando adquirir un número aleatorio.".
      2. Muestra por pantalla el mensaje "Intentando conectar con el servidor: *nombre* con dirección IP: *dirección* para obtener un número aleatorio entre 1 y 1000. "
      3. Muestra por pantalla el mensaje devuelto por el servidor.
      4. Analiza el valor devuelto:
         a. Si es el mismo valor que el seleccionado por el usuario:
            i. El programa muestra por pantalla el mensaje "¡Valor acertado por el servidor!, Terminando programa".
            ii. Envía al servidor el mensaje "FIN" y termina.
         b. Si no es el valor seleccionado por el usuario:
            i. Se repite el proceso.

16. Repetir el problema "14" pero con las modificaciones necesarias para que después de terminar de atender a un cliente, siga disponible para atender a otros clientes.
    a. El servidor **se cerrará** o bien después de atender a 3 clientes o bien cuando el usuario administrador escriba por pantalla "FIN".

SOCKETS MULTICAST

17. Crear un servicio en red mediante una aplicación Java cliente y una aplicación Java servidor que tenga la siguiente funcionalidad:
    a. **Emisor**:
        i. Muestra los siguientes datos propios:
            1. Nombre.
            2. Dirección IP.
            3. Puerto del socket propio.
        ii. Pide al usuario administrador que introduzca la dirección IP y puerto del grupo multicast 1, al que le envía la fecha y hora 1 vez por cada 5 segundos.
        iii. Pide al usuario administrador que introduzca la dirección IP y puerto del grupo multicast 2, al que le envía un mensaje de la fortuna aleatorio de entre 10 opciones diferentes, también cada 5 segundos.
        iv. Realiza esto 10 veces.
        v. Muestra por pantalla mensaje de terminación.
    b. **Receptor**:
        i. Muestra los siguientes datos propios:
            1. Nombre.
            2. Dirección IP.
        ii. Pide al usuario administrador que introduzca la dirección IP y puerto del grupo multicast al que se quiere suscribir.
        iii. Muestra por pantalla los mensajes recibidos.
        iv. Se cierra cuando el usuario teclea “FIN”.
        v. Muestra mensaje de terminación.

ENVÍO DE OBJETOS A TRAVÉS DE SOCKETS

TCP

18. Crear un servicio en red que conste de las siguientes clases Java:
    a. **Servidor**:
        i. Pide al administrador el número de puerto para escuchar peticiones.
        ii. Está a la espera de una petición de un cliente (indicado por pantalla).
        iii. Cuando el cliente se conecta:
            1. Muestra por pantalla que se ha producido una conexión, así como los datos del cliente (dirección IP, puerto y nombre).
            2. Recibe del cliente un objeto rectángulo.
            3. Devuelve al cliente los valores del área y el perímetro.
            4. Queda a la espera de un nuevo envío del cliente.
            5. Recibe del cliente un objeto rectángulo nuevo.
            6. Devuelve al cliente el área y perímetro totales del conjunto de los rectángulos recibidos.
        iv. Realiza 5 iteraciones.
    b. **Cliente**:
        i. Saluda al usuario.
        ii. Pide al usuario la dirección IP o nombre del servidor.
        iii. Pide al usuario el puerto de escucha del servidor.
        iv. Indica al usuario que se va a enviar un rectángulo.
        v. Pide al usuario el ancho del rectángulo.
        vi. Pide al usuario el alto del rectángulo.
        vii. Envía el rectángulo.
        viii. Queda a la espera de la recepción del área y perímetro acumulados.
        ix. Realiza esto 5 veces.
    c. **Objeto**: representará un rectángulo, y tendrá métodos que permitan establecer y obtener tanto el largo como el ancho, así como el área y el perímetro.

19. Basándose en el ejercicio anterior, modificar:
    a. El Servidor almacena el área acumulada en un fichero de texto, y se lo envía al cliente.
    b. El cliente muestra al usuario el valor almacenado en el fichero.

UDP

20. Crear un servicio en red que conste de las siguientes clases Java:
    a. **Servidor**:
        i. Pide al administrador el número de puerto para escuchar peticiones.
        ii. Está a la espera de una petición de un cliente (indicado por pantalla).
        iii. Cuando el cliente se conecta:
            1. Muestra por pantalla que se ha producido una conexión, así como los datos del cliente (dirección IP, puerto y nombre).
            2. Recibe del cliente un objeto rectángulo.
            3. Devuelve al cliente los valores del área y el perímetro.
            4. Queda a la espera de un nuevo envío del cliente.
            5. Recibe del cliente un objeto rectángulo nuevo.
            6. Devuelve al cliente el área y perímetro totales del conjunto de los rectángulos recibidos.
        iv. Realiza 5 iteraciones.
    b. **Cliente**:
        i. Saluda al usuario.
        ii. Pide al usuario la dirección IP o nombre del servidor.
        iii. Pide al usuario el puerto de escucha del servidor.
        iv. Indica al usuario que se va a enviar un rectángulo.
        v. Pide al usuario el ancho del rectángulo.
        vi. Pide al usuario el alto del rectángulo.
        vii. Envía el rectángulo.
        viii. Queda a la espera de la recepción del área y perímetro acumulados.
        ix. Realiza esto 5 veces.
    c. **Objeto**: representará un rectángulo, y tendrá métodos que permitan establecer y obtener tanto el largo como el ancho, así como el área y el perímetro.

21. Basándose en el ejercicio anterior, modificar:
    a. El Servidor almacena el área acumulada en un fichero de texto, y se lo envía al cliente.
    b. El cliente muestra al usuario el valor almacenado en el fichero.

SERVICIOS PARA MÚLTIPLES CLIENTES SIMULTÁNEAMENTE

TCP

22. Crear un servicio en Java que devuelva a los clientes la **fecha y hora en el país indicado por el cliente**, el cual será pasado como parámetro al ejecutar el programa.

23. Crear un servicio en Java que envía a los clientes conectados **un refrán cada 10 segundos**.

24. Crear un **servidor de ficheros** en Java que permita a los clientes:
    a. Enviar un fichero cualquiera al servidor para almacenarlo.
    b. Preguntar al servidor por los ficheros que tiene en la carpeta de compartición, verlos, y poder descargar cualquiera de ellos.

    Para ello el alumno deberá de desarrollar una interfaz de usuario de texto (TUI) que ofrezca un menú de acciones persistente de modo que para salir del programa el usuario tenga que seleccionar la opción para dicha acción:
    1. *Visualizar ficheros en el servidor.*
    2. *Descargar un fichero del servidor.*
    3. *Subir un fichero al servidor.*
    4. *Seleccionar o cambiar de servidor.*
    5. *Salir del programa.*

    Si todavía no está conectado a ningún servidor las opciones 1 a 3 deben mostrar que es necesario estar conectado.

25. Crear un servicio en Java en el cual el cliente manda una **cadena de caracteres** (la cantidad es seleccionada por el usuario cliente), y el **servidor la adivinará** generando combinaciones hasta que obtiene una cadena igual.
    Importante: el programa dejará elegir al cliente *dos tipos de método* para adivinar:
    a. Comparando carácter a carácter.
    b. Comparando la cadena entera.

26. Crear un servicio en Java que permita a varios usuarios **chatear entre ellos en una sala única**, teniendo en cuenta lo siguiente:

a. Al iniciarse el cliente, pide al usuario el nombre de usuario para mostrar en el chat, así como la dirección IP o nombre del servidor.
b. La conversación es almacenada en un fichero de texto en el servidor indicando para cada mensaje:
    i. Usuario/fechayhora/direcciónIP/mensaje.
c. Los mensajes de cada usuario son enviados como objeto.
d. El chat tiene que estar representado en el servidor y clientes por un objeto sincronizado en el que se almacena:
    i. Usuarios como array.
    ii. Número de usuarios.
    iii. Tiempo transcurrido.
e. El cliente da la opción al usuario de salir del chat pulsando una combinación de teclas.
f. En el cliente se debe mostrar para cada mensaje recibido: Usuario/fechayhora/direcciónIP/mensaje.
g. No olvidar que el mensaje del propio usuario tiene que quedar en el log de la pantalla.
h. El servidor permite hasta 20 usuarios en la sala de chat.

27. Crear un servicio en Java que permita a varios usuarios **chatear entre ellos de manera individual**, pero siendo un servicio **centralizado**, teniendo en cuenta lo siguiente:
    a. Basarse en los requerimientos del ejercicio anterior.
    b. Las conversaciones y el estado de la sesión pasan a través del servidor.
    c. El servidor es el que almacena el chat, y reenvía a cada uno de los clientes los nuevos mensajes.
    d. El servidor puede gestionar hasta 1000 conversaciones.

28. Crear un servicio en Java que permita a varios usuarios **chatear entre ellos de manera individua**l, de manera **descentralizada**, teniendo en cuenta lo siguiente:
    a. Basarse en los requerimientos del ejercicio anterior.
    b. Las conversaciones y el estado de la sesión *son gestionados directamente por los clientes* sin necesidad de la intermediación del servidor una vez realizada la conexión.
    c. Los usuarios se conectan al servidor con un nombre de usuario, e indicando el nombre de usuario de la persona con la que quieren hablar, y quedan a la espera hasta que esa persona se conecta.

UDP

29. Convertir los programas anteriores en su respectiva versión UDP.