**ACTIVIDAD 8**

**Enunciado**

Realizar un programa con sockets UDP que establezca un intercambio de información entre un servidor y un único cliente. El intercambio consistirá en que el cliente le enviará al servidor un número y el servidor le devolverá al cliente dicho número elevado al cubo.

Con respecto al **cliente**:

* El nombre de su clase principal será Cliente
* Suponiendo que el número enviado al servidor sea el 4, el resultado esperado de su ejecución será el siguiente:

Esperando respuesta...

Esperando respuesta...: el cubo de + 4 es 64

Adiós…

* Enviará al puerto 12346 del servidor (localhost) el número del que se quiere obtener su valor elevado al cubo
* El número a enviar al servidor no se capturará por pantalla ni se pasará al programa a través un argumento, sino que se definirá dentro del programa, teniendo en cuenta que su rango de valores oscilará entre -128 y +127
* El socket usado para el envío del datagrama al servidor, usará el puerto local 34568
* La frase “Esperando respuesta…” se visualizará justo después de haber enviado el datagrama al servidor
* La frase “Esperando respuesta….: el cubo de…” se visualizará justo después de haber recibido el datagrama del servidor
* La frase “Adiós…” se visualizará justo antes de realizar el cierre del socket

Con respecto al **servidor**:

* El nombre de su clase principal será Servidor
* El resultado esperado de su ejecución será el siguiente, teniendo en cuenta que la dirección IP variará en función del equipo desde el que se ejecute el programa servidor:

Esperando datagrama...

Vamos a calcular el cubo de: 4

IP de origen: /192.168.56.1

Puerto de origen: 34568

Enviamos el resultado...64

Adiósss

* El puerto por el que escuchará las peticiones del cliente será el 12346
* La frase “Esperando datagrama…” se visualizará justo antes de instanciar un objeto de la clase DatagramPacket
* La frase “Vamos a calcular el cubo…” y las dos siguientes se visualizarán después de haber recibido el datagrama desde el cliente
* La frase “Enviamos el resultado…” se visualizará después de haber enviado al cliente el datagrama a través del correspondiente socket
* La frase “Adiósss” se visualizará justo antes de realizar el cierre del socket

Recomendaciones a la hora de realizar las **pruebas** de esta actividad:

* Iniciar la ejecución del servidor antes que la del cliente
* Variar de puerto si surgen problemas

**Solución**

**Cliente.java**

**import java.io.IOException;**

**import java.net.DatagramPacket;**

**import java.net.DatagramSocket;**

**import java.net.InetAddress;**

**import java.net.SocketException;**

**import java.net.UnknownHostException;**

**import java.nio.ByteBuffer;**

**public class Cliente**

**{**

**public static void main (String[] args)**

**{**

**int puerto = 12346;**

**byte buffer[] = new byte[1024];**

**byte bufferRecibo[] = new byte[1024];**

**try**

**{**

**InetAddress destino = InetAddress.getLocalHost();**

**int randomNum = (int)(-128 + Math.random() \* 256);**

**ByteBuffer bbEnvio = ByteBuffer.allocate(4);**

**bbEnvio.putInt(randomNum);**

**buffer = bbEnvio.array();**

**DatagramPacket envio = new DatagramPacket(buffer, buffer.length, destino, puerto);**

**DatagramSocket socket = new DatagramSocket(34568);**

**socket.send(envio);**

**System.out.println("Esperando respuesta...");**

**DatagramPacket recibo = new DatagramPacket(bufferRecibo, bufferRecibo.length);**

**socket.receive(recibo);**

**ByteBuffer bbRecibo = ByteBuffer.wrap(bufferRecibo);**

**System.out.println("Esperando respuesta...: el cubo de " + randomNum + " es " + bbRecibo.getInt());**

**System.out.println("Adiós...");**

**socket.close();**

**}**

**catch (UnknownHostException e)**

**{**

**System.out.println(e.getMessage());**

**}**

**catch (SocketException e)**

**{**

**System.out.println(e.getMessage());**

**}**

**catch (IOException e)**

**{**

**System.out.println(e.getMessage());**

**}**

**}**

**}**

**Servidor.java**

**import java.io.IOException;**

**import java.net.DatagramPacket;**

**import java.net.DatagramSocket;**

**import java.nio.ByteBuffer;;**

**public class Servidor {**

**public static void main (String[] args) {**

**int puerto = 12346;**

**byte buffer[] = new byte[1024];**

**try {**

**DatagramSocket socket = new DatagramSocket(puerto);**

**System.out.println("Esperando datagrama...");**

**DatagramPacket recibo = new DatagramPacket(buffer, buffer.length);**

**socket.receive(recibo);**

**ByteBuffer bbRecibo = ByteBuffer.wrap(buffer);**

**int numRecibido = bbRecibo.getInt();**

**System.out.println("Vamos a calcular el cubo de: " + numRecibido);**

**System.out.println("IP de origen: " + recibo.getAddress().getHostAddress());**

**System.out.println("Puerto de origen: " + recibo.getPort());**

**int resultado = (int)(Math.pow(numRecibido, 2));**

**ByteBuffer bbEnvio = ByteBuffer.allocate(4);**

**bbEnvio.putInt(resultado);**

**byte bufferResultado[] = bbEnvio.array();**

**DatagramPacket envio = new DatagramPacket(bufferResultado, bufferResultado.length, recibo.getAddress(), 34568);**

**socket.send(envio);**

**System.out.println("Enviamos el resultado...");**

**System.out.println("Adiósss");**

**socket.close();**

**}**

**catch (IOException e)**

**{**

**System.out.println(e.getMessage());**

**}**

**}**

**}**

**Consola.java**

import java.io.BufferedInputStream;

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

public class Consola {

public static byte leeByte() {

BufferedReader in=new BufferedReader(new InputStreamReader(System.in));

byte x = 0;

try {

x = Byte.valueOf(in.readLine());

} catch (NumberFormatException | IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

return x;

}

public static short leeShort() {

BufferedReader in=new BufferedReader(new InputStreamReader(System.in));

short x = 0;

try {

x = Short.valueOf(in.readLine());

} catch (NumberFormatException | IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

return x;

}

public static int leeInt() {

BufferedReader in=new BufferedReader(new InputStreamReader(System.in));

int x = 0;

try {

x = Integer.valueOf(in.readLine());

} catch (NumberFormatException | IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

return x;

}

public static long leeLong() {

BufferedReader in=new BufferedReader(new InputStreamReader(System.in));

long x = 0;

try {

x = Long.valueOf(in.readLine());

} catch (NumberFormatException | IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

return x;

}

public static float leeFloat() {

BufferedReader in=new BufferedReader(new InputStreamReader(System.in));

float x = 0;

try {

x = Float.valueOf(in.readLine());

} catch (NumberFormatException | IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

return x;

}

public static double leeDouble() {

BufferedReader in=new BufferedReader(new InputStreamReader(System.in));

double x = 0;

try {

x = Double.valueOf(in.readLine());

} catch (NumberFormatException | IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

return x;

}

public static char leeChar(){

char car = 0;

java.io.BufferedInputStream b = new BufferedInputStream(System.in);

try {

car=(char)b.read();

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

return car;

}

public static String leeString() {

BufferedReader in=new BufferedReader(new InputStreamReader(System.in));

String str = null;

try {

str = in.readLine();

} catch (IOException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

return str;

}

}

**Resolución de problemas**

**CRITERIOS DE CORRECCIÓN**

1. Nivel de ajuste a la nomenclatura de objetos definida en el enunciado de la actividad y nivel de ajuste al almacenamiento de la información requerida en las subcarpetas definidas para la actividad y unidad didáctica correspondientes (15%).
2. Hacer lo que se indica en el enunciado (55%).
3. Claridad del código Java (15%).
4. Documentación del código Java (15%).
5. Explicación detallada de cómo se han resuelto problemas que hayan podido surgir en la realización de la actividad (hasta un 15% adicional sobre la nota obtenida).   
   Dicha explicación se incluirá al principio de los archivos de código fuente Java mediante líneas de comentarios.   
   La primera línea de comentarios sólo incluirá lo siguiente:  
    // Resolución de problemas.
6. Variables no usadas (-15%).
7. Librerías no usadas (-15%).