**ACTIVIDAD 10**

**Enunciado**

Realizar un programa servidor que espere la llegada de un datagrama de un cliente. El cliente le enviará un objeto Tenista que previamente habrá instanciado. El servidor modificará los datos del objeto Tenista y se los enviará de vuelta al cliente. El programa cliente visualizará los datos del objeto Tenista tanto cuando los envía como cuando los recibe. El programa servidor, a su vez, visualizará los datos del objeto Tenista tanto cuando los recibe como cuando los envía modificados.

Con respecto al objeto **Tenista**:

* El nombre de su clase pública será Tenista
* Dispondrá de dos atributos llamados *apellido* (tipo de datos cadena de caracteres) y *altura* (tipo de datos número entero)
* El constructor de la clase gestionará ambos atributos (apellido y altura)
* Dispondrá de dos métodos para obtener el valor de los atributos apellido y altura. Sus nombres serán *getApellido* y *getAltura*
* Dispondrá de dos métodos para modificar el valor de los atributos apellido y altura. Sus nombres serán *setApellido* y *setAltura*

Con respecto al **cliente**:

* El nombre de su clase principal será Cliente
* Después de instanciar un objeto de la clase Tenista con los valores de “del Potro” para el apellido y “198” para la altura, enviará al puerto 12348 del servidor (localhost) un datagrama con el objeto Tenista recién creado
* El socket UDP utilizado para el envío del datagrama utilizará el puerto 34567
* Una vez recibido del servidor el objeto Tenista modificado, se procederá a su visualización
* El resultado esperado de su ejecución será el siguiente:

Envío el objeto: del Potro 198

Esperando datagrama.......

He recibido el objeto: Karlovic 208

Fin del cliente

* La frase “Envío el objeto: del Potro 198” se visualizará justo después de haber enviado el datagrama al servidor
* La frase “Esperando datagrama….…” se visualizará justo antes de haber recibido el datagrama del servidor
* La frase “He recibido el objeto: …” se visualizará después de haber recibido del servidor el objeto Tenista modificado
* La frase “Fin del cliente” se visualizará justo antes de realizar el cierre del socket

Con respecto al **servidor**:

* El nombre de su clase principal será Servidor
* El puerto por el que escuchará las peticiones del cliente será el 12348
* Después de recibir el datagrama con el objeto Tenista desde el cliente, modificará los valores de los atributos de la clase Tenista de la siguiente forma: el apellido pasará a ser “Karlovic” y la altura “208”
* Una vez modificado el objeto Tenista, se escribirá dicho objeto en un stream y se enviará al cliente a través de un socket UDP
* 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.......

Recibo el objeto: del Potro 198

IP de origen: /192.168.56.1

Puerto de origen: 34567

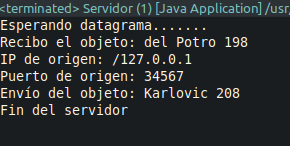
Envío el objeto: Karlovic 208

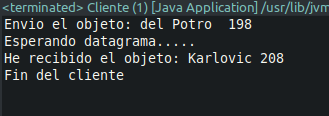
Fin del servidor

* La frase “Esperando datagrama….…” se visualizará justo antes de haber recibido el datagrama del cliente
* La frase “Recibo el objeto: del Potro 198” se visualizará justo después de haber recibido el datagrama del cliente
* La frase “Envío el objeto: Karlovic 208” se visualizará después de haber modificado los 2 atributos del objeto Tenista: apellido y altura
* La frase “Fin del servidor” 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

****

****

**Cliente.java**

import java.io.ByteArrayInputStream;

import java.io.ByteArrayOutputStream;

import java.io.IOException;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.net.DatagramPacket;

import java.net.DatagramSocket;

import java.net.InetAddress;

import java.net.UnknownHostException;

public class Cliente {

public static void main (String[] args) {

try {

//Instanciamos el tenista, determinamos el puerto

Tenista tenista = new Tenista("del Potro", 198);

int puerto = 12348;

InetAddress destino = InetAddress.getLocalHost();

//Instanciamos los outputStream

ByteArrayOutputStream baos = new ByteArrayOutputStream();

ObjectOutputStream oos = new ObjectOutputStream(baos);

oos.writeObject(tenista);

byte bufferEnviar[] = baos.toByteArray();

baos.close();

oos.close();

//Instanciamos los datagrams

DatagramSocket socket = new DatagramSocket(34567);

DatagramPacket enviar = new DatagramPacket(bufferEnviar, bufferEnviar.length, destino, puerto);

//enviamos

socket.send(enviar);

System.out.println("Envio el objeto: " + tenista.getApellido() + " " + tenista.getAltura());

//Recivimos respuesta del servidor

byte bufferRecibir[] = new byte[bufferEnviar.length];

DatagramPacket recibir = new DatagramPacket(bufferRecibir, bufferRecibir.length);

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

socket.receive(recibir);

//Instanciamos los inputStream

ByteArrayInputStream bais = new ByteArrayInputStream(bufferRecibir);

ObjectInputStream ois = new ObjectInputStream(bais);

Tenista tenistaRecibir = (Tenista)ois.readObject();

bais.close();

ois.close();

//Escribimos los mensajes y cerramos

System.out.println("He recibido el objeto: " + tenistaRecibir.getApellido() + " " + tenistaRecibir.getAltura());

System.out.println("Fin del cliente");

socket.close();

}

catch (IOException e){

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

}

catch (ClassNotFoundException e) {

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

}

}

}

**Servidor.java**

import java.io.ByteArrayInputStream;

import java.io.ByteArrayOutputStream;

import java.io.IOException;

import java.io.ObjectInputStream;

import java.io.ObjectOutputStream;

import java.net.DatagramPacket;

import java.net.DatagramSocket;

import java.net.InetAddress;

public class Servidor {

public static void main (String[] args) {

try {

//Instanciamos puerto y datagams

int puerto = 12348;

byte bufferRecibir[] = new byte[1024];

DatagramPacket recibir = new DatagramPacket(bufferRecibir, bufferRecibir.length);

DatagramSocket socket = new DatagramSocket(puerto);

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

socket.receive(recibir);

//Instanciamos los inputStream

ByteArrayInputStream bais = new ByteArrayInputStream(bufferRecibir);

ObjectInputStream ois = new ObjectInputStream(bais);

Tenista tenista = (Tenista)ois.readObject();

bais.close();

ois.close();

//Mostramos mensajes

System.out.println("Recibo el objeto: " + tenista.getApellido() + " " + tenista.getAltura());

System.out.println("IP de origen: " + recibir.getAddress());

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

//Creamos al tenista

tenista.setApellido("Karlovic");

tenista.setAltura(208);

System.out.println("Envío del objeto: " + tenista.getApellido() + " " + tenista.getAltura());

//Instanciamos los outputStream

ByteArrayOutputStream baos = new ByteArrayOutputStream();

ObjectOutputStream oos = new ObjectOutputStream(baos);

oos.writeObject(tenista);

byte bufferEnviar[] = baos.toByteArray();

baos.close();

oos.close();

//Enviamos la informacion

DatagramPacket enviar = new DatagramPacket(bufferEnviar, bufferEnviar.length, InetAddress.getLocalHost(), 34567);

socket.send(enviar);

//Cerramos

System.out.println("Fin del servidor");

socket.close();

}

catch (IOException e) {

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

}

catch (ClassNotFoundException e) {

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

}

}

}

**Tenista.java**

import java.io.Serializable;

public class Tenista implements Serializable

{

//clase Tenista con atributos y constructor

private static final long serialVersionUID = 1L;

private String apellido;

private int altura;

public Tenista(String apellido, int altura) {

this.apellido = apellido;

this.altura = altura;

}

//getters y setters

public String getApellido() {

return apellido;

}

public void setApellido(String apellido) {

this.apellido = apellido;

}

public int getAltura() {

return altura;

}

public void setAltura(int altura) {

this.altura = altura;

}

}

**REQUISITOS**

**REQUISITOS**

1. Entregar las soluciones de las actividades en un único archivo comprimido donde tendremos:
   1. Un archivo de extensión .odt o .docx que contendrá
      1. Una copia del código del programa
      2. Una captura de pantalla del resultado de la ejecución del programa
   2. Los archivos \*.java empleados
2. El único archivo con la soluciones contendrá el nombre original de la actividad, seguido de un guion bajo y del primer apellido del alumno.

**CRITERIOS DE CORRECCIÓN**

1. Cumplimiento de los requisitos (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%).