3.7 ENVÍO DE OBJETOS A TRAVES DE SOCKETS

Hasta ahora hemos estado intercambiando cadenas de caracteres entre programas cliente y servidor. Pero los stream soportan diversos tipos de datos como son los bytes, los tipos de datos primitivos, caracteres localizados y objetos.

En este apartado veremos como se pueden intercambiar objetos entre programas emisor y receptor o entre programas cliente y servidor usando sockets.

3.7.1. OBJETOS EN SOCKETS TCP

Las clases ObjectInputStream y ObjectOutputStream nos permiten enviar objetos a través de sockets TCP. Utilizaremos los métodos readObject() para leer el objeto del stream y writeObject() para escribir el objeto al stream. Usaremos el constructor que admite un InputStream y un OutputStream. Para preparar el flujo de salida para escribir objetos escribimos:

```
ObjectOutputStream outObjeto =
new ObjectOutputStream(socket.getOutputStream());
```

Para preparar el flujo de entrada para leer objetos escribimos:

```
ObjectInputStream inObjeto = new
ObjectInputStream(socket.getInputStream());
```

Las clases a la que pertenecen estos objetos deben implementar la interfaz Serializable. Por ejemplo, sea la clase Persona con 2 atributos, nombre y edad, 2 constructores y los métodos get y set correspondientes:

```
import java.io.Serializable;
@SuppressWarnings("serial")
public class Persona implements Serializable {
String nombre;
int edad;
public Persona(String nombre, int edad) f
                    super();
this.nombre = nombre;
this.edad = edad;
}
public Persona() {super();}

public String getNombre() {return nombre;}
public void setNombre(String nombre) {this.nombre = nombre;}
public int getEdad() (return edad;}
public void setEdad(int edad) {this.edad = edad;}
```

```
}
```

Podemos intercambiar objetos Persona entre un cliente y un servidor usando sockets TCP. Por ejemplo el programa servidor crea un objeto Persona, dándole valores y se lo envía al programa cliente, el programa cliente realiza los cambios oportunos en el objeto y se lo devuelve modificado al servidor. El programa servidor es el siguiente:

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

public class TCPObjetoServidorl
public static void main(String[] arg) throws I0Exception,
ClassNotFoundException {
int numeroPuerto = 6000;// Puerto
ServerSocket servidor = new ServerSocket(numeroPuerto);

System.out.println("Esperando al cliente…") ;
Socket cliente = servidor.accept();

// Se prepara un flujo de salida para objetos
ObjectOutputStream outObjeto = new ObjectOutputStream(
cliente.getOutputStream());

// Se prepara un objeto y se envia
Persona per = new Persona("Juan", 20) ;
outObjeto.writeObject(per); //enviando objeto
        System.out.println("Envio: " +per.getNombre () +"*"+per.getEdad();

// Se obtiene un stream para leer objetos
ObjectInputStream inObjeto = new ObjectInputStream(
cliente.getInputStream());
Persona dato = (Persona) inObjeto.readObject();
System.out.println("Recibo: "+dato.getNombre()+"*"+dato.getEdad());

//CERRAR STREAMS Y SOCKETS
outObjeto.close();
inObjeto.close();
cliente.close();
```

```
servidor.close();
}
}//
```

El programa cliente es el siguiente:

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

public class TCPObjetoCliente1{
public static void main(Stringt] arg) throws I0Exception,
ClassNotFoundException
String Host = "localhost";
int Puerto = 6000;//puerto remoto

System.out.println("PROGRAMA CLIENTE INICIADO ") ;
Socket cliente = new Socket(Host, Puerto);

//Flujo de entrada para objetos
ObjectInputStream perEnt = new ObjectlnputStream(
cliente.getInputStream());

//Se recibe un objeto
Persona dato = (Persona) perEnt.readObject();
System.out.println("Recibo: "+dato.getNombre()+"*"+dato.getEdad());

//Modifico el objeto
dato.setNombre("Juan Ramos");
dato.setEdad(22);

//FLUJO DE salida para objetos
ObjectOutputStream perSal = new ObjectOutputStream(
cliente.getOutputStream());

// Se envia el objeto
perSal.writeObject(dato);
System.out.println("Envio: "+dato.getNombre()+"*"+dato.getEdad());
```

```
//CERRAR STREAMS Y SOCKETS
perEnt.close();
perSal.close();
cliente.close();
}
}//..
```

Cuando usamos un bucle para enviar objetos se recomienda utilizar el método reset() antes de enviar el objeto por el stream, de esta manera se ignorara el estado de cualquier objeto ya escrito en el stream. Ejemplo:

outObjeto.reset();

outObjeto.writeObject(per);

ACTIVIDAD 3.7

Crea una clase Java llamada números que defina 3 atributos, uno de ellos entero, y los otros dos de tipo long:

int numero;

long cuadrado;

long cubo;

Define un constructor con parámetros y otro sin parámetros. Define los métodos get y set de los atributos. Crea un programa cliente que introduzca por teclado un numero e inicialice un objeto Numeros, el atributo numero debe contener el numero introducido por teclado. Debe enviar ese objeto al programa servidor. El proceso se repetirá mientras el número introducido por teclado sea mayor que 0.

Crea un programa servidor que reciba un objeto Numeros. Debe calcular el cuadrado y el cubo del atributo numero y debe enviar el objeto al cliente con los cálculos realizados, el cuadrado y el cubo en sus atributos respectivos. El cliente recibirá el objeto y visualizara el cuadrado y el cubo del número introducido por teclado. El programa servidor finalizará cuando el numero recibido en el objeto Numeros sea menor o igual que 0.

Controlar posibles errores, por ejemplo si ejecutamos el cliente y el servidor no está iniciado, o si estando el servidor ejecutándose ocurre algún error en el cliente, o este finaliza inesperadamente, etc.