CLIENTE QUE ENVIA UNA PETICION AL SERVIDOR.
ENVIA Y RECIBE INFORMACION CON EL SERVIDOR

```
/*
 * El cliente se debe conectar al servidor y enviar numeros del 1 hasta el 10000
 * para encontrar la clave aleatoria que el servidor ha generado. Lo intenta
 * hasta que el acierte y entonces recibe del servidor un mensaje concreto. En
 * cuando llegue ese mensaje en concreto, entonces el cliente para de enviar numeros
 */
package Ejercicio1;

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

/**
 *
 * @author Usuario
 */
public class Cliente_1 {

   static final int puerto = 6000;
   static final String host = "localhost";
   int numero = 1;
   boolean conseguido = false;

   /*
   Constructor del Cliente_1
    */
   public Cliente_1() {
      try {
         Socket sCliente = new Socket(host, puerto);// El cliente se conecta al servidor por el
puerto
         /*
         Una vez establecida la conexion entre el cliente y el servidor, hay que enviar y recibir
         datos a través del socket, para lo que se usa un STREAM
          */

         //Defino los Stream para el flujo de salida del CLiente al Servidor
         OutputStream salida = sCliente.getOutputStream();
         DataOutputStream flujoSalida = new DataOutputStream(salida);

         //Defino los stream para el flujo de entrada desde el Servidor al Cliente_1
         InputStream entrada = sCliente.getInputStream();
         DataInputStream flujoEntrada = new DataInputStream(entrada);

         System.out.println("PROGRAMA CLIENTE INICIADO....");
         while (!conseguido) {
            flujoSalida.writeUTF("" + numero);//defino el flujo de salida y envio numeros para
probar

            if (flujoEntrada.readUTF().toString().startsWith("Clave incorrecta")) {
```

```
                System.out.println("He intentado como clave el valor "
                        + numero + " y me dice: Clave Incorrecta. Llevas "
                        + numero + " intentos");
                numero++;

            } else {
                conseguido = true;
                System.out.println("He intentado como clave el valor "
                        + numero + " y me dice: Has acertado la clave en  "
                        + numero + " intentos");
            }
        }
        flujoEntrada.close();
        flujoSalida.close();

        sCliente.close();// Cierro el socket

    } catch (IOException e) {
        System.out.println(e.getMessage());
    }
  }

  public static void main(String[] arg) {
    new Cliente_1();

  }
}
```

SERVIDOR QUE PERMITE UNA SOLA CONEXIÓN A UN CLIENTE. RECIBE Y ENVIA INFORMACION CON EL CLIENTE SIN CORAR LA CONEXIÓN.

```
/*
El servidor genera un numero aleatorio entre 1 y 10000 y atiende la peticion de 1 cliente (solo
1) que le envia 1000 intentos para descrifrar el numero.
Si lo consigue manda un mensaje y cierra la conexion. Si no lo consigue manda otro mensaje y
espera hasta que se llegue a los 1000 numeros. Si a los 1000 numeros no se ha conseguido
cierra la conexion y el cliente deja de enviar numeros
 */
package Ejercicio1;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.logging.Level;
import java.util.logging.Logger;


public class Servidor_1 {


   public static void main(String[] args) {
      int numeroPuerto = 6000;
      int clave = generarClave();
      int intento = 0;
      boolean descifrada = false;
      int numIntentos = 0;
      ServerSocket servidor = null;
      Socket clienteConectado = null;
      InputStream entrada = null;
      OutputStream salida = null;
      DataInputStream flujoEntrada = null;
      DataOutputStream flujoSalida = null;
      String mensaje = "";
      int intentosMax = 1000;//numero de intentos máximos de conexiones

      try {
         servidor = new ServerSocket(numeroPuerto);

         System.out.println("Esperando al cliente...");

         clienteConectado = servidor.accept();

         //CREO FLUJO DE ENTRADA DEL CLIENTE
         entrada = clienteConectado.getInputStream();
         flujoEntrada = new DataInputStream(entrada);
```

```
            //CREO FLUJO DE SALIDA AL CLIENTE
            salida = clienteConectado.getOutputStream();
            flujoSalida = new DataOutputStream(salida);
        } catch (IOException ex) {
            Logger.getLogger(Servidor_1.class.getName()).log(Level.SEVERE, null, ex);
        }
        System.out.println("Se conecta un cliente que intenta descifrar la clave.");
        System.out.println("Clave: " + clave);
        while (!descifrada) {

            try {
                //EL CLIENTE ME ENVÍA UNA PRUEBA DE CLAVE
                intento = Integer.parseInt(flujoEntrada.readUTF());
//                System.out.println("Clave recibida. Valor de la clave: "+intento);

                numIntentos++;
                //ENVÍO LA RESPUESTA ANTE EL INTENTO
                if (clave == intento) {
                    flujoSalida.writeUTF("Has acertado la clave en " + numIntentos + " intentos.");
                    descifrada = true;
                } else {
                    flujoSalida.writeUTF("Clave incorrecta. Llevas " + numIntentos + " intentos.");
                }

            } catch (IOException ex) {
                Logger.getLogger(Servidor_1.class.getName()).log(Level.SEVERE, null, ex);
            }
        }

        System.out.println("El cliente ha descifrado la clave en " + numIntentos + " intentos.");

        try {
            //CERRAR STREAMS Y SOCKETS
            entrada.close();
            flujoEntrada.close();
            salida.close();
            flujoSalida.close();
            clienteConectado.close();
            servidor.close();
        } catch (IOException ex) {
            Logger.getLogger(Servidor_1.class.getName()).log(Level.SEVERE, null, ex);
        }
    }

    public static int generarClave() {
        int numero = (int) (Math.random() * 9999) + 1;
        return numero;
    }
}
```

SERVIDOR QUE PERMITE UN SOLO CLIENTE, SE COMUNICA Y CORTA LA CONEXIÓN PASADO UN NUMERO DE INTENTOS

```
/*
El nuevo Servidor solo puede admitir 1000 peticiones de un Cliente.
En caso de recibir 1000 peticiones y no averiguar la clave, el Servidor se cerrará.
 */
package Ejercicio2;

import Ejercicio1.*;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.logging.Level;
import java.util.logging.Logger;

public class Servidor_2 {

   static final int intentosMax = 1000;//numero de intentos máximos de conexiones

   /**
    * @param args the command line arguments
    */
   public static void main(String[] args) {
      int numeroPuerto = 6000;
      int clave = generarClave();
      int probarNum = 0;
      boolean descifrada = false;
      int numIntentos = 0;

      /*
      Declaracion de servidor y socket
       */
      ServerSocket servidor = null;
      Socket clienteConectado = null;

      /*
      Declaracion de flujos de entrada
       */
      InputStream entrada = null;
      DataInputStream flujoEntrada = null;

      /*
      Declaracion de flujos de salida
       */
      OutputStream salida = null;
      DataOutputStream flujoSalida = null;
```

```
        String mensaje = "";

        try {
            servidor = new ServerSocket(numeroPuerto);//El servidor escucha el puerto

            System.out.println("Esperando al cliente...");

            clienteConectado = servidor.accept();//acepto la conexión del cliente

            //CREO FLUJO DE ENTRADA DEL CLIENTE
            entrada = clienteConectado.getInputStream();
            flujoEntrada = new DataInputStream(entrada);

            //CREO FLUJO DE SALIDA AL CLIENTE
            salida = clienteConectado.getOutputStream();
            flujoSalida = new DataOutputStream(salida);
        } catch (IOException ex) {
            Logger.getLogger(Servidor_2.class.getName()).log(Level.SEVERE, null, ex);
        }
        System.out.println("Se conecta un cliente que intenta descifrar la clave.");
        System.out.println("Clave: " + clave);

        /*
        Mientras la clave no esté descubierta y no se hayan superado el numero
        de intentos, se sigue probando
         */
        do {
            try {
                //RECIBO EL NUMERO QUE EL CLIENTE ME ENVÍA
                probarNum = Integer.parseInt(flujoEntrada.readUTF());

                numIntentos++;
                //ENVÍO LA RESPUESTA ANTE EL INTENTO

                if (clave == probarNum) {
                    flujoSalida.writeUTF("Clave CORRECTA. Llevas " + numIntentos + " intentos.");
                    descifrada = true;
                } else {//Si no ha encontrado el numero en el intento, mando este mensaje
                    flujoSalida.writeUTF("Clave incorrecta. Llevas " + numIntentos + " intentos");
                }

            } catch (IOException ex) {
                Logger.getLogger(Servidor_2.class.getName()).log(Level.SEVERE, null, ex);
            }

        } while (descifrada == false && (numIntentos < intentosMax));
        if (!descifrada) {
            try {
                flujoSalida.writeUTF("Game Over. No has conseguido descrifrar la clave en " +
intentosMax + " intentos");

            } catch (IOException ex) {
```

```
            Logger.getLogger(Servidor_2.class.getName()).log(Level.SEVERE, null, ex);
        }

        try {
            //CERRAR STREAMS Y SOCKETS
            entrada.close();
            flujoEntrada.close();
            salida.close();
            flujoSalida.close();
            clienteConectado.close();
            servidor.close();
        } catch (IOException ex) {
            Logger.getLogger(Servidor_2.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
  }

  public static int generarClave() {
    int numero = (int) (Math.random() * 9999) + 1;
    return numero;
  }
}
```

CLIENTE QUE SE COMUNICA CON UN SERVIDOR ENVIANDO Y RECIBIENDO INFORMACIÓN Y QUE EN UN MOMENTO DADO PUEDE CORTAR LA CONEXIÓN

/*
De esta forma, observarás que tu Cliente, que en el ejercicio 1 siempre acababa acertando la clave,
ahora la acertará o no, en función de la clave aleatorio que se generé y el rango de números que el cliente pregunte (por lo que solo tendrás que cambiar las salidas por consola ante las respuestas del Servidor y que el Cliente deje de preguntar cuando el Servidor te avise de que se va a cerrar).
Aquí tienes un ejemplo de cómo podría quedar la ejecución de tu Cliente con el nuevo Servidor (localhost, puerto 6000):
PROGRAMA CLIENTE INICIADO...
He intentado como clave el valor 1 y me dice: Clave incorrecta. Llevas 1 intentos.
He intentado como clave el valor 2 y me dice: Clave incorrecta. Llevas 2 intentos.

...

He intentado como clave el valor 998 y me dice: Clave incorrecta. Llevas 998 intentos.
He intentado como clave el valor 999 y me dice: Clave incorrecta. Llevas 999 intentos.
Game over. No he conseguido descifrar la clave.

*/

```
package Ejercicio2;

import Ejercicio1.*;
import java.net.*;
import java.io.*;


public class Cliente_2 {

   static final int puerto = 6000;
   static final String host = "localhost";
   int numero = 1;
   boolean conseguido, finalizado = false;
   String mensajeRecibido = "";

   /*
   Constructor del Cliente_2
    */
   public Cliente_2() {
      try {
         Socket sCliente = new Socket(host, puerto);// El cliente se conecta al servidor por el
puerto
         /*
         Una vez establecida la conexion entre el cliente y el servidor, hay que enviar y recibir
         datos a través del socket, para lo que se usa un STREAM
          */
```

```
        //Defino los Stream para el flujo de salida del CLiente al Servidor
        OutputStream salida = sCliente.getOutputStream();
        DataOutputStream flujoSalida = new DataOutputStream(salida);

        //Defino los stream para el flujo de salida desde el Servidor al Cliente_2
        InputStream entrada = sCliente.getInputStream();
        DataInputStream flujoEntrada = new DataInputStream(entrada);

        System.out.println("PROGRAMA CLIENTE INICIADO....");

        while (!conseguido && !finalizado) {
          //System.out.println("envio el numero "+numero);
          flujoSalida.writeUTF("" + numero);//defino el flujo de salida y envio numeros para
probar
          mensajeRecibido = flujoEntrada.readUTF();
          if (mensajeRecibido.startsWith("Clave incorrecta")) {
            System.out.println("He intentado como clave el valor "
                + numero + " y me dice: " + mensajeRecibido);
            numero++;
          }
          if (mensajeRecibido.startsWith("Clave CORRECTA")) {
            System.out.println("He intentado como clave el valor "
                + numero + " y me dice: " + mensajeRecibido);
            conseguido = true;

          }
          if (mensajeRecibido.startsWith("Game")) {
            System.out.println(mensajeRecibido);
            finalizado = true;

          }
        }
        //CERRAR STREAMS Y SOCKETS
        salida.close();
        flujoSalida.close();
        entrada.close();
        flujoEntrada.close();
        sCliente.close();// Cierro el socket

      } catch (IOException e) {
        System.out.println(e.getMessage());
      }

    }

    public static void main(String[] arg) {
      new Cliente_2();

    }
}
```

CLASE SERVIDOR DE LA QUE VAMOS A LANZAR HILOS PARA ATENDER A PETICIONES DE CLIENTES. ESTA CLASE ES LA QUE DEFINE EL SERVIDOR

```
/*
 Clase servidor de la que vamos a instanciar hasta 100 objetos según las peticiones de los clientes
 */
package prueba3;

import java.io.*;
import java.net.*;
import java.util.*;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
public class Servidor_3 extends Thread {

   static final int intentosMax = 2;//1000;
   static final int puerto = 6000;
   static final String host = "localhost";

   Socket sCliente;
   int numCliente = 1;

   InputStream entrada = null;
   DataInputStream flujoEntrada = null;

   OutputStream salida = null;
   DataOutputStream flujoSalida = null;

   /*
   Constructor
    */
   public Servidor_3(Socket sCliente, int numCliente) {
      this.sCliente = sCliente;
      this.numCliente = numCliente;

   }

   @Override
   public void run() {

      int probarNum = 1;
      int numIntentos = 0;
      int clave = 5;//generarClave();
      boolean descifrada = false;

      boolean conseguido, finalizado = false;

      System.out.println("Iniciado el hilo del servidor: " + this.numCliente);
      System.out.println("La clave es: " + clave);
```

```
        do {
            try {
                //Defino los Stream para el flujo de salida del Cliente al Servidor_3
                this.salida = sCliente.getOutputStream();
                this.flujoSalida = new DataOutputStream(salida);

                // Flujo de entrada a través del cual recibiremos información desde el proceso cliente
                this.entrada = sCliente.getInputStream();
                this.flujoEntrada = new DataInputStream(entrada);

            } catch (IOException ex) {
                Logger.getLogger(Servidor_3.class.getName()).log(Level.SEVERE, null, ex);
            }

            try {
                //RECIBO EL NUMERO QUE EL CLIENTE ME ENVÍA
                probarNum = Integer.parseInt(this.flujoEntrada.readUTF());

                numIntentos++;
                //ENVÍO LA RESPUESTA ANTE EL INTENTO

                if (clave == probarNum) {//si el cliente ha descifrado la clave
                    this.flujoSalida.writeUTF("Clave CORRECTA. Llevas " + numIntentos + " intentos.");
                    descifrada = true;
                    System.out.println(
                            "El cliente " + this.numCliente + " ha descifrado la clave en " + numIntentos + "
intentos");

                    System.out.println(
                            "Hilo servidor:" + this.numCliente + " Fin de la conexión con el cliente");
                    this.sCliente.close();
                }
                if (numIntentos<intentosMax && descifrada==false){//Si no ha encontrado el numero
en el intento, mando este mensaje
                    this.flujoSalida.writeUTF("Clave incorrecta. Llevas " + numIntentos + " intentos");
                }
                if (descifrada=false && numIntentos==intentosMax){
                    System.out.println("El cliente " + this.numCliente + " ha agotado el numero de
intentos sin descifrar la clave");
                this.flujoSalida.writeUTF("Clave incorrecta. Llevas " + numIntentos + "
intentos\n"+"Game Over. No has conseguido descrifrar la clave en " + intentosMax + "
intentos");
                }

            } catch (IOException ex) {
                Logger.getLogger(Servidor_3.class.getName()).log(Level.SEVERE, null, ex);
            }

        } while (descifrada == false && (numIntentos < intentosMax));


            try {
```

```
            // Cerramos la comunicación con el cliente
            this.flujoSalida.close();
            this.flujoEntrada.close();
            this.sCliente.close();

        } catch (SocketException ex) {
            System.out.printf("Error de socket: %s\n", ex.getMessage());
        } catch (IOException ex) {
            System.out.printf("Error de E/S: %s\n", ex.getMessage());
        }

        System.out.println(
              "Hilo servidor:" + this.numCliente + " Fin de la conexión con el cliente");
    }

    public static int generarClave() {
        int numero = (int) (Math.random() * 20)+1;//9999) + 1;
        return numero;
    }
}
```

CLIENTE QUE SE VA A LANZAR SOBRE UN SEVIDOR QUE ATENDERA VARIAS PETICIONES. SE COMINICA CON EL SERVIDOR.

```
package prueba3;

import java.io.*;
import java.net.Socket;
import java.util.*;
import java.util.logging.Level;
import java.util.logging.Logger;

public class Cliente_3 extends Thread {

   public static void main(String[] args) {

      Socket cliente = null;
      int puerto = 6000;

      int numeroQueMando = 1;//

      //Defino los Stream para el flujo de entrada del Cliente al Servidor
      InputStream entrada = null;
      DataInputStream flujoEntrada = null;

      String mensajeRecibido = "";
      String mensajeSalida = "";

      //Defino los Stream para el flujo de salida del Cliente al Servidor
      OutputStream salida = null;
      DataOutputStream flujoSalida = null;

      System.out.println("PROGRAMA CLIENTE INICIADO ...");
      try {
         cliente = new Socket("localhost", puerto);

         entrada = cliente.getInputStream();
         flujoEntrada = new DataInputStream(entrada);

         salida = cliente.getOutputStream();
         flujoSalida = new DataOutputStream(salida);

         /*
         CUIDADO!!! Este while tiene que controlar bien para que las salidas
         y entradas de mensajes entre el servidor y el cliente esten sincronizados
         */
         while (mensajeRecibido.startsWith("Clave CORRECTA")==false &&
               mensajeRecibido.startsWith("Game")==false){

            mensajeSalida = (""+numeroQueMando);
            flujoSalida.writeUTF(mensajeSalida);//defino el flujo de salida y envio numeros para
probar
```

```
            mensajeRecibido = flujoEntrada.readUTF();
            /*
            Si recibo del servidor el mensaje que empieza por "Clave incorrecta...."
            entonces intento con el siguiente numero
            */
            if (mensajeRecibido.startsWith("Clave incorrecta")) {
               System.out.println("He intentado como clave el valor "
                     + numeroQueMando + " y me dice: " + mensajeRecibido);
               numeroQueMando++;
            }
            /*
            Si recibo del servidor el mensaje que empieza por "Clave CORRECTA...."
            entonces he conseguido acertar el número
            */
            if (mensajeRecibido.startsWith("Clave CORRECTA")) {
               System.out.println("He intentado como clave el valor "
                     + numeroQueMando + " y me dice: " + mensajeRecibido);
//               conseguido = true;

            }
            /*
            Si recibo del servidro el mensaje que empieza por "Game..." entonces han terminado
los intentos, no
            he conseguido adivinar la clave y finalizo la busqueda
            */
            if (mensajeRecibido.startsWith("Game")) {
               System.out.println(mensajeRecibido);
//               finalizado = true;

            }
         }


      } catch (IOException ex) {
         ex.printStackTrace();
      }

      //Cierro los Streams y los Sockets
      try {
         entrada.close();
         flujoEntrada.close();
         salida.close();
         flujoSalida.close();
         cliente.close();
      } catch (IOException io) {
         io.printStackTrace();
      }
   }
}
```

CLASE QUE CONTIENE EL MAIN Y QUE VA A LANZAR TANTOS HILOS DE SERVIDOR (HASTA UN LIMITE) COMO PETICIONES POR PARTE DECLIENTES VENGAN

```
/*
 Esta clase lanza hasta 100 servidores por hilos para atender hasta
100 peticiones de clientes a la vez.
 */
package prueba3;

import java.io.IOException;
import java.net.*;
import java.util.ArrayList;

/**
 *
 * @author Usuario
 */
public class MainServidor_3 {

    /**
     * @param args the command line arguments
     */
   public static void main(String[] args) throws InterruptedException {
      int puerto=6000;
      ArrayList<Thread> hilosServidor = new ArrayList();
      //ServerSocket sServidor=null;
      int numClientesConectados = 0;

      ServerSocket servidor = null;
      //Socket clienteConectado = null;

      try {
         // Punto de conexión del servidor (socket en el puerto indicado en la máquina donde se
ejecute el proceso)
         servidor = new ServerSocket(puerto);

         // Mensaje de arranque de la aplicación
         System.out.println("SERVIDOR CONCURRENTE");
         System.out.println("--------------------");
         System.out.println("Servidor iniciado.");
         System.out.println("Escuchando por el puerto: "+ puerto);
         System.out.println("Esperando conexión con cliente.");


         while (numClientesConectados < 100) {

            // Quedamos a la espera ("escuchando") de que se realice una conexión con el socket
de servidor.
            // En el momento en que eso suceda, se aceptará. Mientras tanto, la ejecución queda
aquíe
```

```
            // bloqueada en espera a que se reciba esa petición por parte de un cliente.
            Socket clienteConectado = servidor.accept();

            //Interacción del servidor con un cliente
            System.out.println("Conexión establecida con cliente.");

            // Creamos un nuevo hilo de ejecución para servir a este nuevo cliente conectado
            Servidor_3 hiloServidor = new Servidor_3(clienteConectado, numClientesConectados
+ 1);
            hilosServidor.add(hiloServidor);
            numClientesConectados++;

            // Lanzamos la ejecución de ese nuevo hilo
            hiloServidor.start();
          } // y seguimos "escuchando" a otras posibles peticiones de cliente

          //Cuando finalice cierro el socket del servidor
          servidor.close();

        } catch (SocketException ex) {
          System.out.printf("Error de socket: %s\n", ex.getMessage());
        } catch (IOException ex) {
          System.out.printf("Error de E/S: %s\n", ex.getMessage());
        }

        System.out.println("Fin de ejecución del servidor.");
    }

}
```

ESQUEMAS CLIENTE – SERVIDOR

Servidor (TCP)

1.- Crear conexión:

```
ServerSocket skServidor = new Server(puerto);
```

2.- Crear flujos de entrada y salida de información

```
InputStream entrada = null;
DataInputStream flujoEntrada = null;

OutputStream salida = null;
DataOutputStream flujoSalida = null;

//CREO FLUJO DE ENTRADA DEL CLIENTE
entrada = clienteConectado.getInputStream();
flujoEntrada = new DataInputStream(entrada);

//CREO FLUJO DE SALIDA AL CLIENTE
salida = clienteConectado.getOutputStream();
flujoSalida = new DataOutputStream(salida);
```

3.- Aceptar conexión:

```
Socket sCliente = skServidor.accept();
```

4.- Atiendo peticion del cliente

```
//CREO FLUJO DE ENTRADA DEL CLIENTE
        entrada = clienteConectado.getInputStream();
        flujoEntrada = new DataInputStream(entrada);

        //CREO FLUJO DE SALIDA AL CLIENTE
        salida = clienteConectado.getOutputStream();
        flujoSalida = new DataOutputStream(salida);
```

Hago lo que el servidor tenga que hacer

```
//RECIBO EL NUMERO QUE EL CLIENTE ME ENVÍA
        probarNum = Integer.parseInt(flujoEntrada.readUTF());

        numIntentos++;
//ENVÍO LA RESPUESTA ANTE EL INTENTO

        if (clave == probarNum) {
                flujoSalida.writeUTF("Clave CORRECTA. Llevas " + numIntentos + " intentos.");
                descifrada = true;
        } else {//Si no ha encontrado el numero en el intento, mando este mensaje
        flujoSalida.writeUTF("Clave incorrecta. Llevas " + numIntentos + " intentos");
}
```

5.- Cierro socket y flujos

```
//CERRAR STREAMS Y SOCKETS
        entrada.close();
        flujoEntrada.close();
        salida.close();
        flujoSalida.close();
        clienteConectado.close();
        servidor.close();
```

Cliente (TCP)

```
final int puerto = 6000;
final String host = "localhost";

Socket sCliente = new Socket(host, puerto);// El cliente se conecta al servidor por el puerto
        /*
        Una vez establecida la conexión entre el cliente y el servidor, hay que enviar y recibir
        datos a través del socket, para lo que se usa un STREAM
         */

        //Defino los Stream para el flujo de salida del CLiente al Servidor
        OutputStream salida = sCliente.getOutputStream();
        DataOutputStream flujoSalida = new DataOutputStream(salida);

        //Defino los stream para el flujo de entrada desde el Servidor al Cliente_1
        InputStream entrada = sCliente.getInputStream();
        DataInputStream flujoEntrada = new DataInputStream(entrada);
```

```
while (!conseguido) {
            flujoSalida.writeUTF("" + numero); // EL CLIENTE, LO PRIMERO QUE HACE ES ENVIAR

            if (flujoEntrada.readUTF().toString().startsWith("Clave incorrecta")) {// Y DESPUES
RECIBE LO QUE SEA
                System.out.println("He intentado como clave el valor + numero +.....
```

CIERRO LAS CONEXIONES Y STREAMS

```
flujoEntrada.close();
flujoSalida.close();
sCliente.close();// Cierro el socket
```