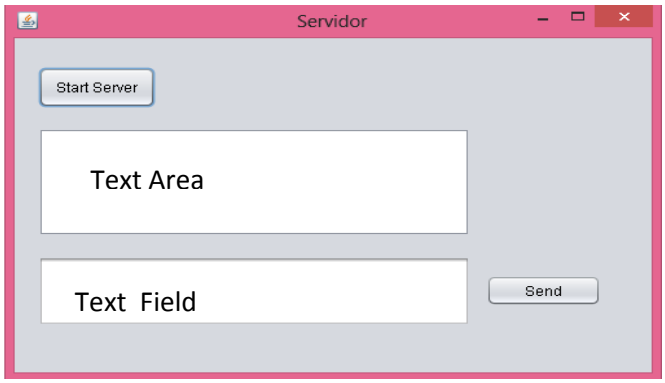
UNIVERSIDAD DE NARIÑO
SISTEMAS DISTRIBUIDOS
GUIA SOCKET EN JAVA – CHAT /CLIENTE-SERVIDOR
PARTE II

Por: José Javier Villalba Romero

Paso 1. Regresamos en Design a la ServerForm para incluir:

1 TextArea { Que sirve para mostrar los mensajes del cliente
1 TextField { Que sirve para redactar el mensaje que se enviará al cliente
1 Boton { Que permite enviar el flujo de datos del servidor al cliente

Forma Servidor

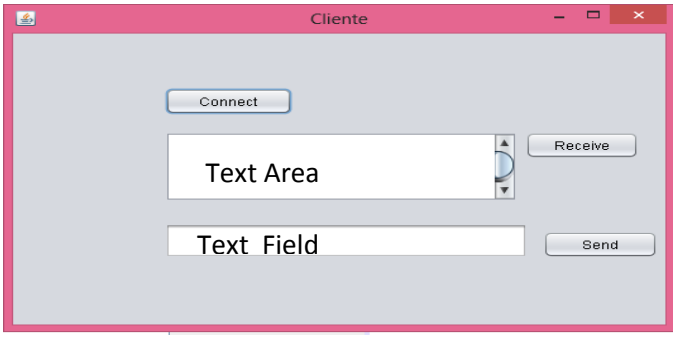


Al Text Area le cambiamos el name: txt_RecMsg
Al Text Field le cambiamos el name: txt_msg
Al Boton Send le cambiamos el name: btn_enviar

Paso 2. Entramos en Design de la ClientForm e incluimos:

1 TextArea { Que sirve para mostrar los mensajes del servidor
1 TextField { Que sirve para redactar el mensaje que se enviará al servidor
1 Botón { Que permite enviar el flujo de datos del cliente al servidor

Forma Cliente



Al Text Area le cambiamos el name: txt_RecMsg
Al Text Field le cambiamos el name: txt_msg
Al Boton Send le cambiamos el name: btn_Recibir

Paso 3. En source del Servidor declaramos las variables dos y dis que controlan en flujo de datos en salida y entrada desde cliente y servidor y las inicializamos con *null*. Esto lo hacemos debajo de las líneas:

Socket client = null;
DataOutputStream dos = null;
DataInputStream dis = null;

Para lo anterior se deben importar las clases que permiten manejar el flujo de datos

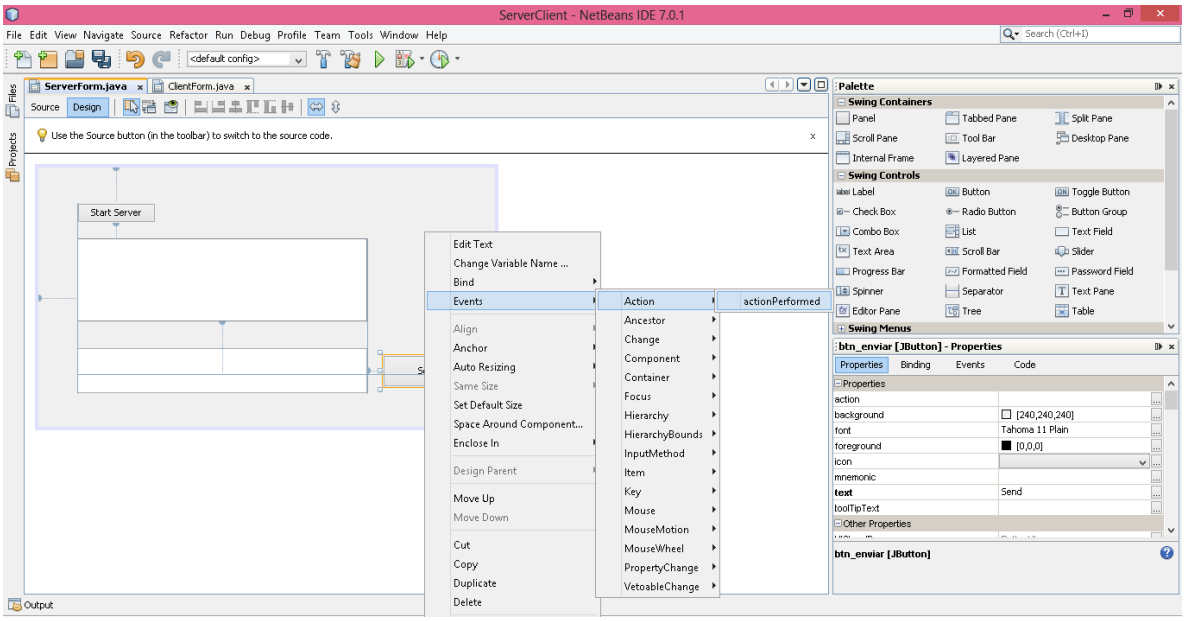
import java.io.DataInputStream;
import java.io.DataOutputStream;

```
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;

/**
 *
 * @author Usuario
 */
public class ServerForm extends javax.swing.JFrame {
ServerSocket server = null;
Socket client = null;
DataOutputStream dos = null;
DataInputStream dis = null;
```

Paso 4. En el Botón Start Server del Servidor entramos con click Derecho + Event + Action + Action Performed y colocamos el flujo de datos a ser enviados del servidor al cliente, para ello, colocamos las siguientes líneas debajo de:

JOptionPane.showMessageDialog(null,"Cliente Aceptado");
dos = new DataOutputStream(client.getOutputStream());




```
private void btn_iniciarActionPerformed(java.awt.event.ActionEvent evt) {
        try {
            server = new ServerSocket(6969);
            client = server.accept();
            JOptionPane.showMessageDialog(null,"Cliente Aceptado");
            dos = new DataOutputStream(client.getOutputStream());
        } catch (IOException ex) {
            JOptionPane.showMessageDialog(null,"Cliente no Disponible");
            Logger.getLogger(ServerForm.class.getName()).log(Level.SEVERE, null, ex);
        }
}
```

Paso 5. Programamos ahora el Botón SEND del Servidor entrando con Click Derecho + Event + Action + Action Performed. Con ello tomamos los datos que están en el txt_msg y los enviamos al cliente. Para ello escribimos las siguientes instrucciones en la clase del btn_enviar y manejando las excepciones respectivas así:

```
private void btn_enviarActionPerformed(java.awt.event.ActionEvent evt) {
        String msg = txt_msg.getText();
         try {
             dos.writeUTF(msg);
         } catch (IOException ex) {
             Logger.getLogger(ServerForm.class.getName()).log(Level.SEVERE, null, ex);
         }
}
```

Paso 6. Regresamos a la ClientForm y en Source declaramos las variables para manejar el flujo de datos hacia el servidor. Esto lo hacemos debajo de la línea

Socket server = null;
DataInpuntStream dis = null;
DataOutputStream dos = null;

Manejamos las excepciones agregando las clases que permiten manejar los flujos de entrada y de salida.

import java.io.DataInputStream;
import java.io.DataOutputStream;

Quedando así:

```
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.Socket;
import java.net.UnknownHostException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JOptionPane;


public class ClientForm extends javax.swing.JFrame {

    Socket server = null;
    DataInputStream dis = null;
    DataOutputStream dos = null;
```

Paso 7. En modo Design de ClientForm agragamos la siguiente línea al botón Connect con Click derecho + Event + Action + Action Performed , debajo de la línea:

JOptionPane.showMessageDialog(null,"Conectado al Servidor");
dis = new DataInputStream(server.getInputStream());

Paso 8. En el evento del Boton Recibir del Cliente colocamos las siguientes líneas:

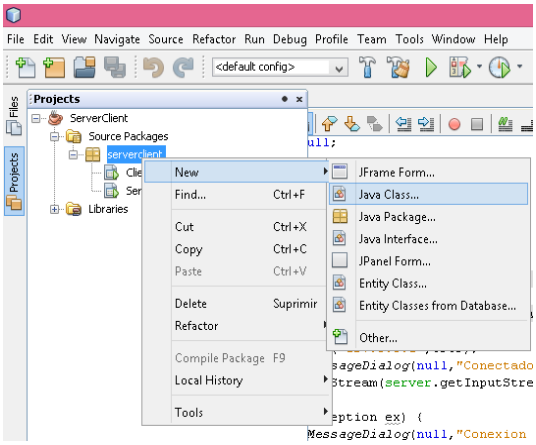
String msg = dis.readUTF();
txt_RecMsg.append("\n" + msg);

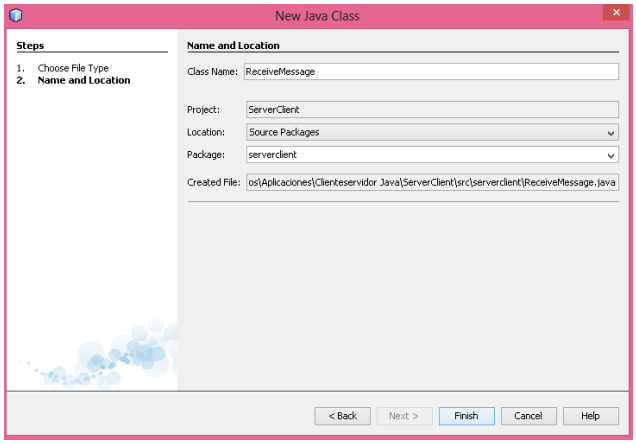
Manejamos las excepciones quedando así:

```
private void btn_RecibirActionPerformed(java.awt.event.ActionEvent evt) {
        try {
            String msg = dis.readUTF();
            txt_RecMsg.append("\n" + msg);
        } catch (IOException ex) {
            Logger.getLogger(ClientForm.class.getName()).log(Level.SEVERE, null, ex);
        }

}
```

Paso 9. Creamos una nueva clase en Package llamado ReceiveMessage que permite manejar los hilos del socket.





A esta clase le agregamos lo siguiente para manejar los hilos:

public class ReceiveMessage extends Thread

Paso 10. En esta Clase escibimos el siguiente código sin olvidar importar estas dos clases:

import java.io.DataInputStream;
import javax.swing.JTextArea;

```
public class ReceiveMessage extends Thread {
    String msg = "  ";
    DataInputStream dis = null;
    JTextArea txt_area = null;
    public ReceiveMessage(DataInputStream d, JTextArea a){
```

```
        this.dis = d;
        this.txt_area = a;
    }
    public void run(){
        while(true)
        {
            try {
                msg = dis.readUTF();
                txt_area.append("\n" +this.getName()+":"+msg);
            } catch (IOException ex) {
                Logger.getLogger(ReceiveMessage.class.getName()).log(Level.SEVERE,    null,
ex);
            }
        }
    }

}
```

```
import java.io.IOException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JTextArea;

public class ReceiveMessage extends Thread {
    String msg = "  ";
    DataInputStream dis = null;
    JTextArea txt_area = null;
    public ReceiveMessage(DataInputStream d, JTextArea a){
        this.dis = d;
        this.txt_area = a;
    }
    public void run(){
        while(true)
        {
            try {
                msg = dis.readUTF();
                txt_area.append("\n" +this.getName()+":"+msg);
            } catch (IOException ex) {
                Logger.getLogger(ReceiveMessage.class.getName()).log(Level.SEVERE, null, ex);
            }
        }
    }

}
```

Paso 11. En el Boton Connect de la ClientForm debajo de la siguiente instrucción

dis = new DataInputStream(server.getInputStream());

Colocamos lo siguiente:

ReceiveMessage clientThread = new ReceiveMessage(dis, txt_RecMsg);
clientThread.setDaemon(true);
clientThread.setName("Server :");
clientThread.start();

```
private void btn_ConectarActionPerformed(java.awt.event.ActionEvent evt) {
        try {
            server = new Socket("127.0.0.1",6969);
            JOptionPane.showMessageDialog(null,"Conectado al Servidor");
            dis = new DataInputStream(server.getInputStream());
            ReceiveMessage clientThread = new ReceiveMessage(dis, txt_RecMsg);
            clientThread.setDaemon(true);
            clientThread.setName("Server :");
            clientThread.start();

        } catch (UnknownHostException ex) {
            JOptionPane.showMessageDialog(null,"Conexion Fallida");
        } catch (IOException ex) {
            JOptionPane.showMessageDialog(null,"Conexion Fallida");
        }
}
```

Paso 12. En el Botón Send de la ClientForm dando click derecho + Event + Action + Action Perfomed, colocamos las siguientes instrucciones, manejando las excepciones con Try y Catch así:

dos.writeUTF(txt_msg.getText());

```
private void btn_enviarActionPerformed(java.awt.event.ActionEvent evt) {
        try {
            dos.writeUTF(txt_msg.getText());
        } catch (IOException ex) {
            Logger.getLogger(ClientForm.class.getName()).log(Level.SEVERE, null, ex);
        }
}
```

Paso 13. En el Botón Connect de la ClientForm debajo de la siguiente instrucción dis = new DataInputStream(server.getInputStream());
colocamos:
dos = new DataOutputStream(server.getOutputStream());

```
private void btn_ConectarActionPerformed(java.awt.event.ActionEvent evt) {
        try {
            server = new Socket("127.0.0.1",6969);
            JOptionPane.showMessageDialog(null,"Conectado al Servidor");
            dis = new DataInputStream(server.getInputStream());
            dos = new DataOutputStream(server.getOutputStream());
            ReceiveMessage clientThread = new ReceiveMessage(dis, txt_RecMsg);
            clientThread.setDaemon(true);
            clientThread.setName("Server :");
            clientThread.start();
```

Paso 14. En el Botón Star Sever de la ServerForm debajo de la línea:

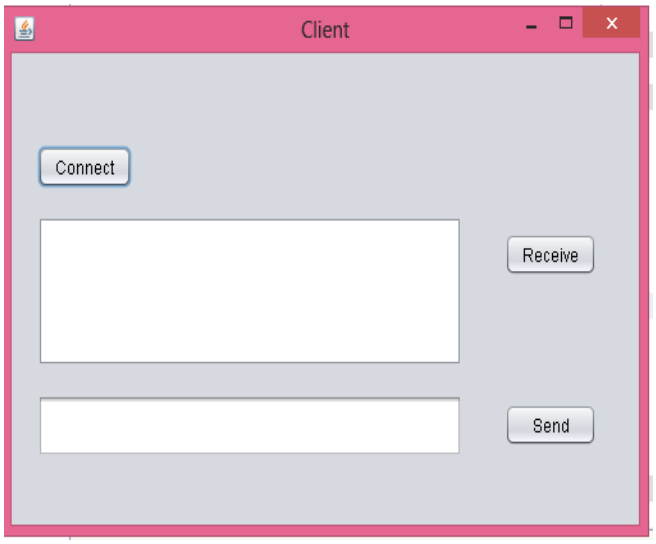
dos = new DataOutputStream(client.getOutputStream());

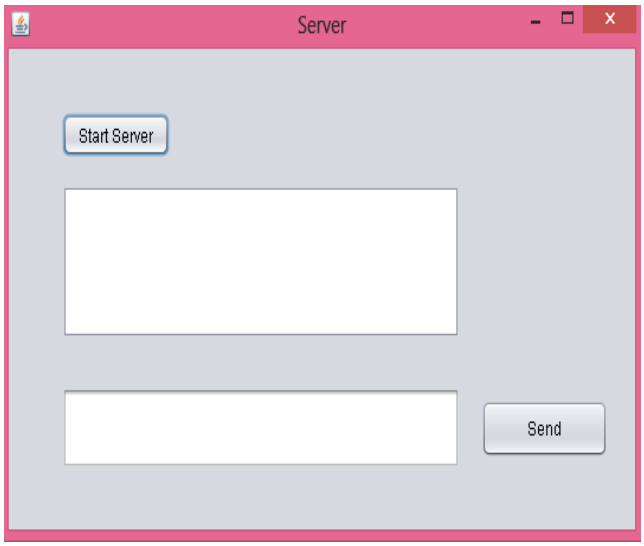
Colocamos lo siguiente:

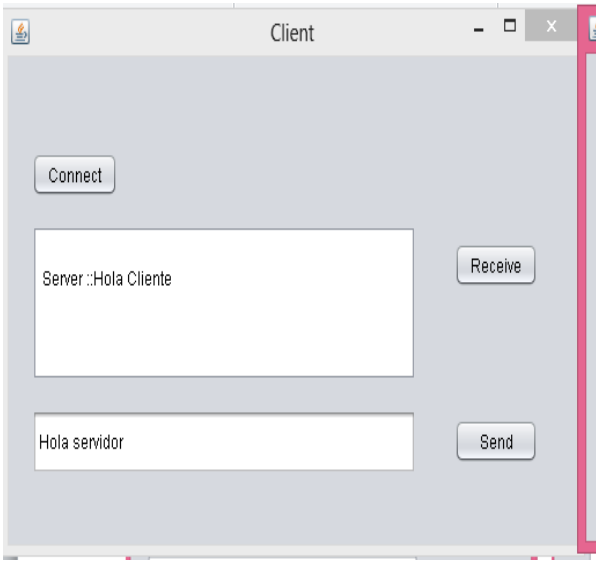
dis = new DataInputStream(client.getInputStream());
ReceiveMessage serverThread = new ReceiveMessage(dis,txt_RecMsg);
serverThread.setDaemon(true);
serverThread.setName("Cliente");
serverThread.start();

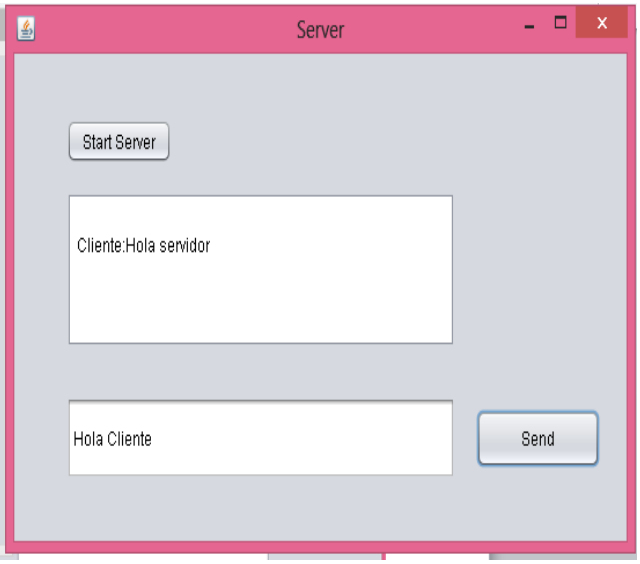
```
private void btn_iniciarActionPerformed(java.awt.event.ActionEvent evt) {
        try {
            server = new ServerSocket(6969);
            client = server.accept();
            JOptionPane.showMessageDialog(null,"Cliente Aceptado");
            dos = new DataOutputStream(client.getOutputStream());
            dis = new DataInputStream(client.getInputStream());
            ReceiveMessage serverThread = new ReceiveMessage(dis,txt_RecMsg);
            serverThread.setDaemon(true);
            serverThread.setName("Cliente");
            serverThread.start();
        } catch (IOException ex) {
            JOptionPane.showMessageDialog(null,"Cliente no Disponible");
            Logger.getLogger(ServerForm.class.getName()).log(Level.SEVERE, null, ex);
        }
}
```

Con lo anterior hemos terminado de crear la aplicación cliente y la servidor, ahora procedemos a correr las dos aplicaciones.


Client
Connect
Receive
Send


Server
Start Server
Send


Client
Connect
Server ::Hola Cliente
Receive
Hola servidor
Send


Server
Start Server
Cliente:Hola servidor
Hola Cliente
Send