

Descripción del equipo y de las herramientas de trabajo:

Descripción del equipo de cómputo:

Esta guía se desarrolló usando un computador de escritorio tipo portátil marca HP, con Windows 10 home, microprocesador Intel Core i5 de 1.8 GHZ 8a generación, 8 GB de memoria RAM unidad de almacenamiento de 250 GB de estado sólido, talla de 16.7 pulgadas, 2 gigas de memoria de vídeo tarjeta nvidia.

Acerca de

El equipo está supervisado y protegido.

[Ver detalles en Seguridad de Windows](#)

Especificaciones del dispositivo

HP Laptop 15-dw0xxx

Nombre del dispositivo

Procesador	Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz 1.80 GHz
RAM instalada	8,00 GB (7,89 GB utilizable)
Id. del dispositivo	
Id. del producto	
Tipo de sistema	Sistema operativo de 64 bits, procesador x64
Lápiz y entrada táctil	La entrada táctil o manuscrita no está disponible para esta pantalla

Copiar

Cambiar el nombre de este equipo

Especificaciones de Windows

Edición	Windows 10 Home
Versión	20H2
Se instaló el	6/04/2021
Compilación del SO	19042.867
Experiencia	Windows Feature Experience Pack 120.2212.551.0

Descripción del software utilizado

Para el desarrollo de esta guía fue necesario Descargar instalar y configurar adecuadamente las siguientes herramientas de software:

- Desarrollo de aplicaciones Java o JDK
 - Java SE Development Kit 8 - Downloads | Oracle Colombia
 - <https://www.oracle.com/co/java/technologies/javase/javase-jdk8-downloads.html>
 - INSTALACIÓN DEL JDK EN WINDOWS 10
 - <https://youtu.be/BoYRcKZbDb0>
- Entorno integrado de desarrollo en Netbeans 12.3
 - [I](#)NSTALACIÓN DE NETBEANS 12 WINDOWS
 - <https://youtu.be/oT1cUI984zU>
- Procesador de texto online Google Documents
- Code Block, complemento para Google Documents
- PlantUML Gizmo, complemento para Google Documents.

Descripción de la aplicación tomada como ejemplo.

Con el fin de simplificar el desarrollo de ejemplo y la explicación del mismo, he optado por realizar una simple aplicación tipo cliente servidor, la cual consta de dos sistemas:

Una aplicación que actúa como servicio y otra aplicación que actúa como cliente

Las características de la aplicación servidor Serán las siguientes:


- Inicia abriendo un puerto lógico de red específico estipulado por el usuario
- Acepta, atiende y procesa solicitudes de conexiones provenientes desde Una o varias instancias de aplicaciones clientes previamente desarrolladas para establecer comunicación con el servidor
- durante el proceso de conexión la aplicación servidor recibe 2 datos (peso y altura) enviado por la aplicación cliente
- la aplicación servidor procesa estos datos y calcula el índice de masa corporal o IMC.
- Como respuesta el servidor envía el IMC y un mensaje a la aplicación cliente que realizó la solicitud.

Características de la aplicación cliente

- Al ejecutarla se despliega una interfaz de usuario con un formulario, el cual solicita al usuario ingresar la dirección IP y el número de puerto utilizados por la aplicación servidor
- El usuario un botón para conectarse con la aplicación servidor.
- como respuesta positiva de la conexión la aplicación despliega un formulario para ingresar los datos correspondientes al cálculo del IMC (peso y altura)
- el usuario pulsa un botón para enviar los datos al servidor el cual realiza el cálculo y devuelve el resultado con respuesta.
- La aplicación cliente muestra el resultado del cálculo enviado por el servidor.

A continuación se presenta boceto las interfaces de ambas aplicaciones:.

GUÍA DE LA APLICACIÓN CLIENTE

	<pre>@startsalt {+ . Cliente IMC . {/ Conexión Cálculo IMC } . {+ . <I> IP Servidor: " . <I> Puerto Servidor: " . } . .. { [Cancelar<&circle-x>] </pre>
---	---

Guía básica de Desarrollo de sistemas distribuidos usando Socket TCP y Java.

Autor: John Carlos Arrieta Arrieta

	<pre> [Conectar<&account-login>] } } @endsalt</pre>
	<pre>@startsalt {+ . Servidor IMC . {/ Conexion Calculo IMC } == . { <I> PESO: " . <I> ALTURA: " . } -- Resultado: <I>0.0 .. { . [Cancelar<&circle-x>] [Cálculo<&account- login>] } } @endsalt</pre>

GUÍA DE LA APLICACIÓN SERVIDOR

Servidor IMC

ConexiónLog de Conexiones

Puerto de Servicio:

Estado: Detenido

CancelarIniciar

```
@startsalt
{+
.
Servidor IMC
.
  {/<b> Conexión | Log de Conexiones
  }
  ==
.
.
  {
    <B>Puerto de Servicio: | " "
    .
  }
  --
.
  <B>Estado: </b><I> Detenido
  .
  ..
  {
    .
    [Cancelar<&circle-x>] |
    [Iniciar <&account-login>]
  }
}
@endsalt
```

Servidor IMC

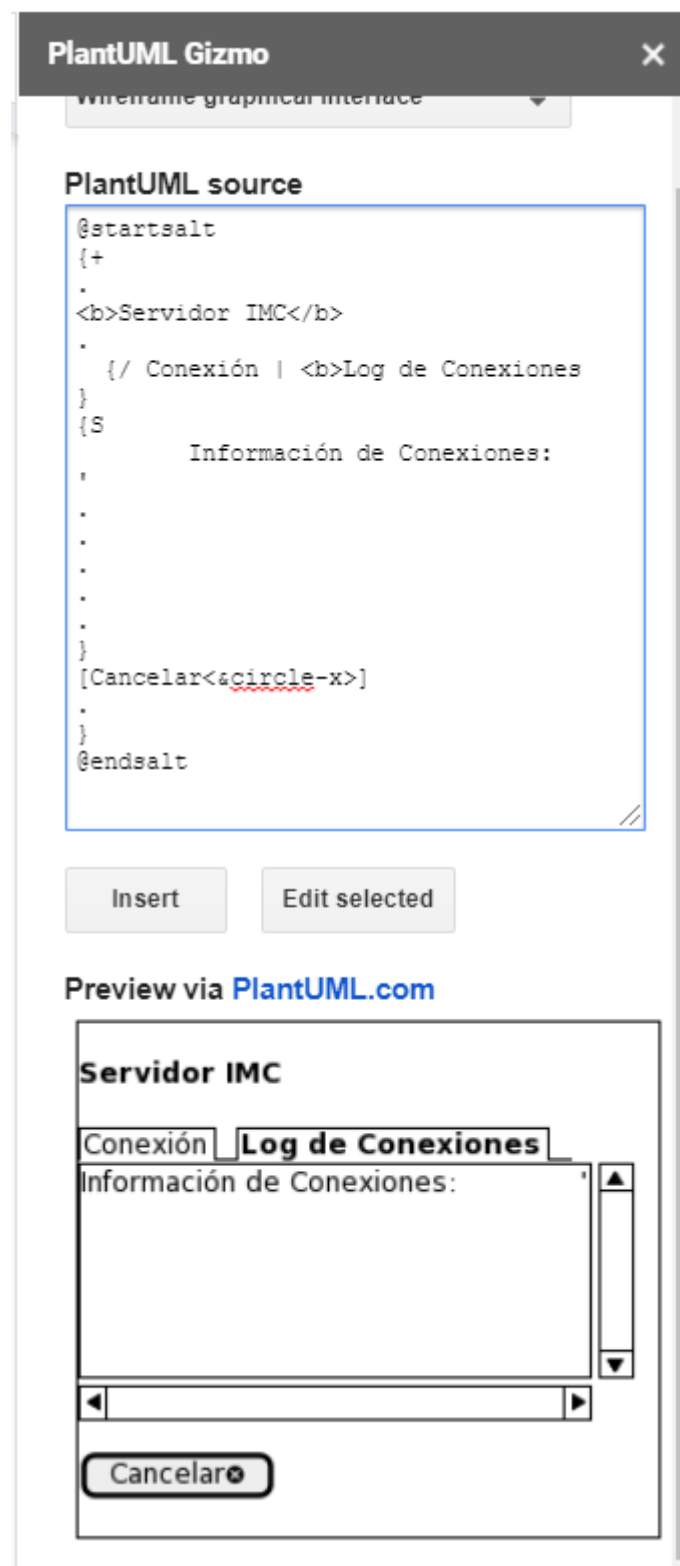
ConexiónLog de Conexiones

Información de Conexiones:

Limpiar

```
@startsalt
{+
.
<b>Servidor IMC</b>
.
  {/ Conexión | <b>Log de Conexiones  }
  {S
    Información de Conexiones:
    '
    .
    .
    .
    .
    .
  }
  [Cancelar<&circle-x>]
.
}
@endsalt
```

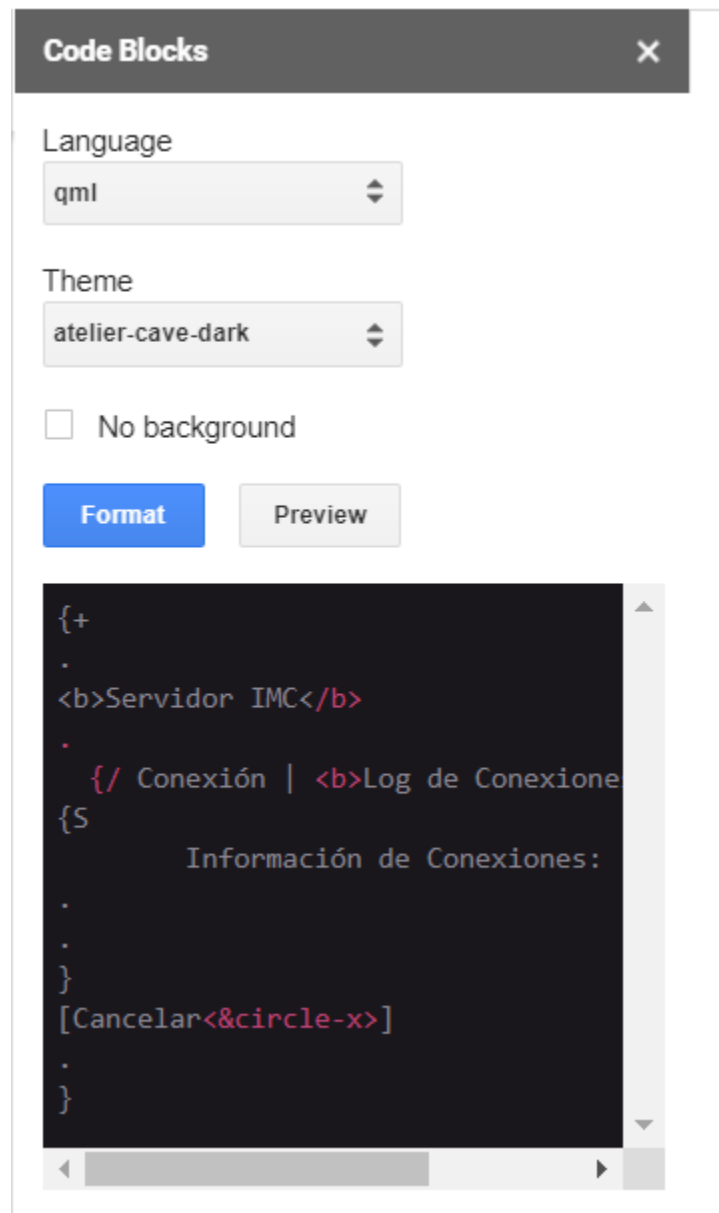
Ejemplo el uso del complemento PlantUML GIZMO



Ejemplo del uso del complemento CODEBLOCKS

Guía básica de Desarrollo de sistemas distribuidos usando Socket TCP y Java.

Autor: John Carlos Arrieta Arrieta



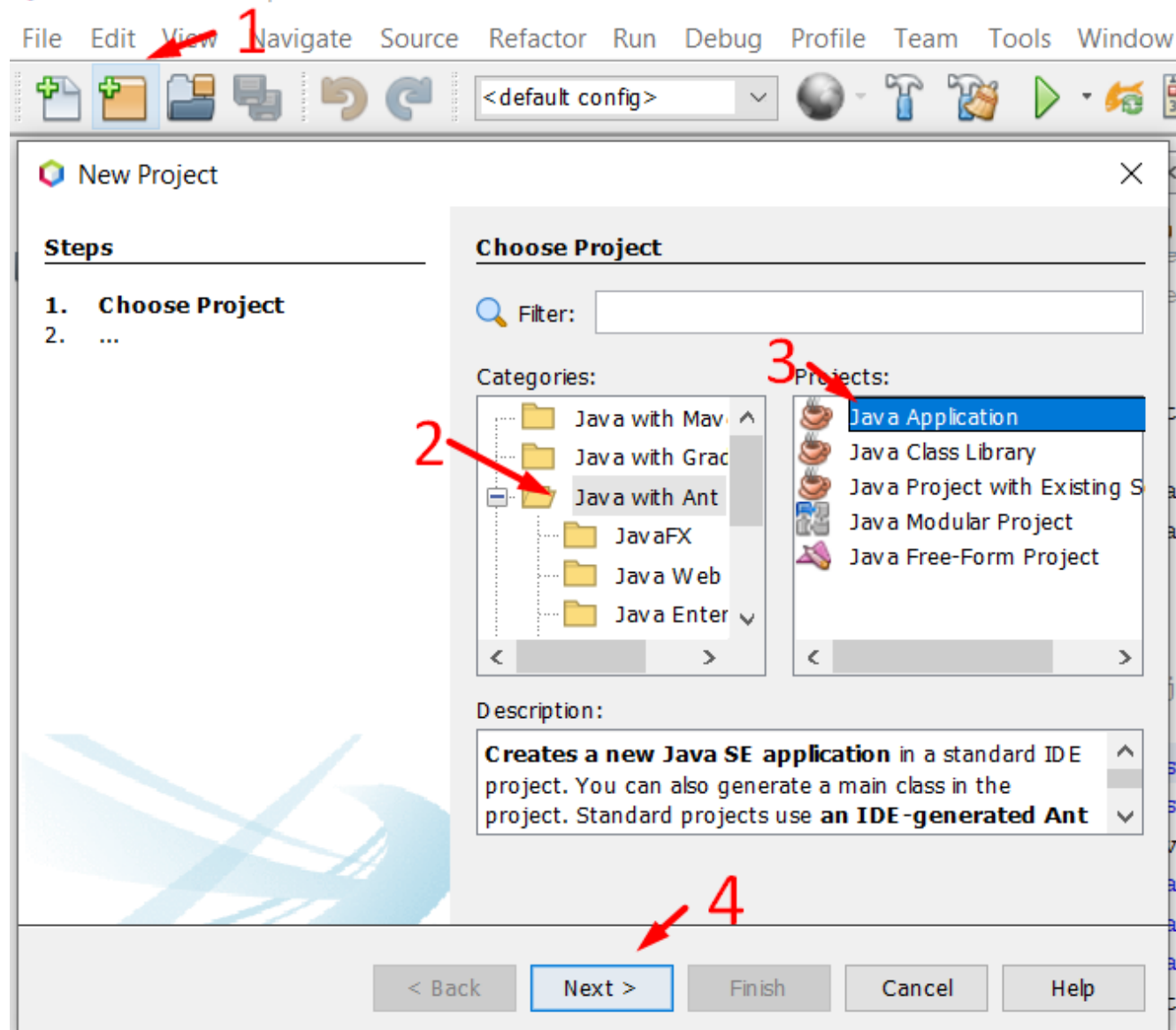
Guía básica de Desarrollo de sistemas distribuidos usando Socket TCP y Java.

Autor: John Carlos Arrieta Arrieta

CREAR EL PROYECTO EN NETBEANS

Creamos el proyecto en netbeans seleccionando Java with Ant y Java Application

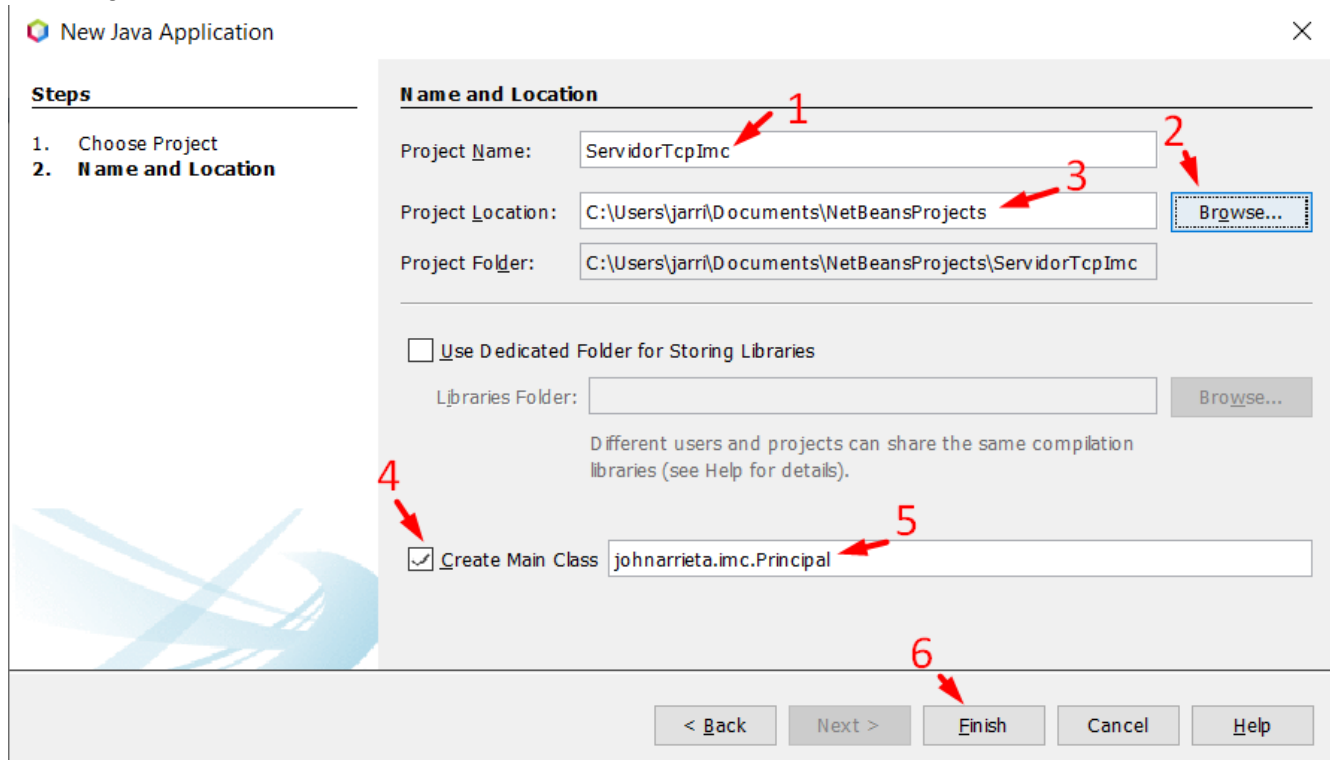
sumawscliente - Apache NetBeans IDE 12.2



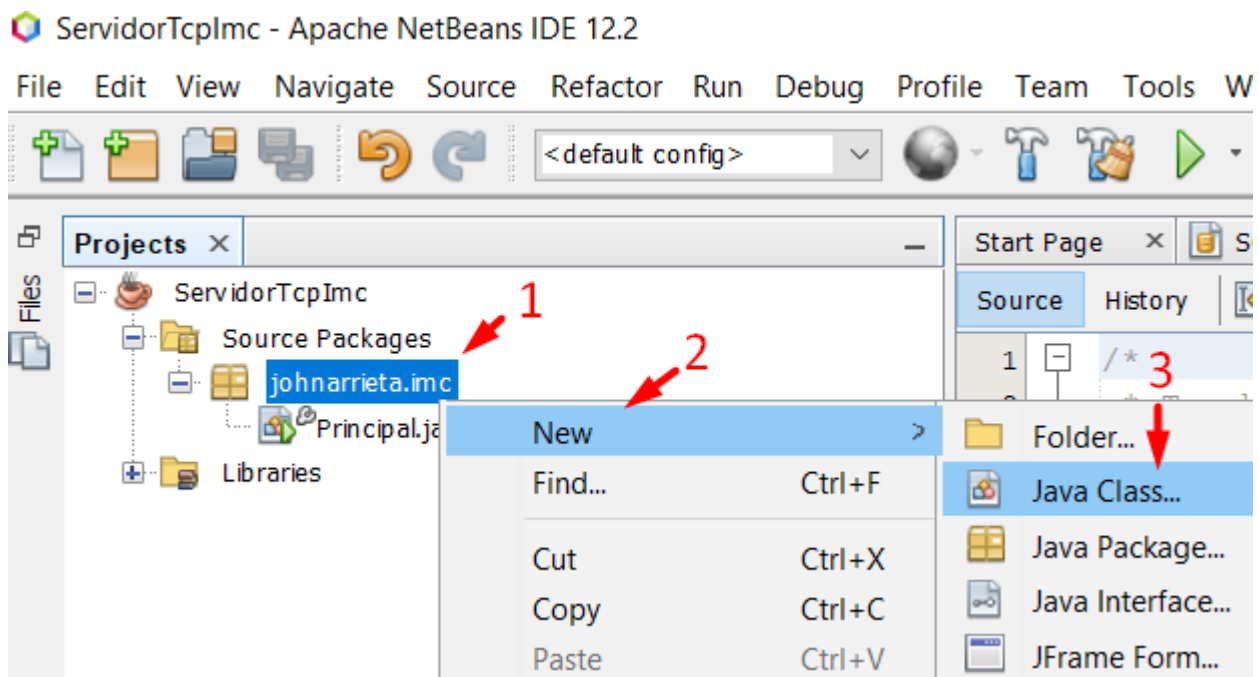
Guía básica de Desarrollo de sistemas distribuidos usando Socket TCP y Java.

Autor: John Carlos Arrieta Arrieta

Le colocamos un nombre al proyecto y seleccionamos el paquete junto con la clase principal y el damos guardar



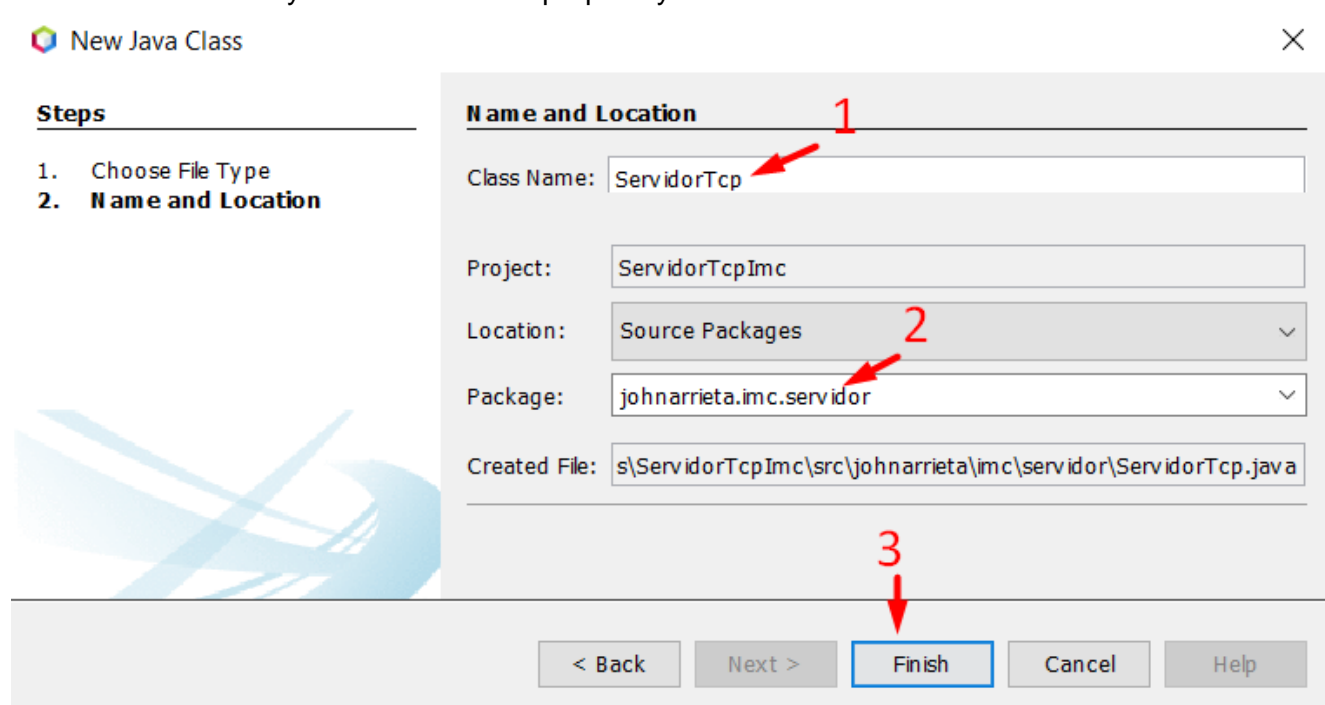
Vamos a crear una clase dentro de este paquete



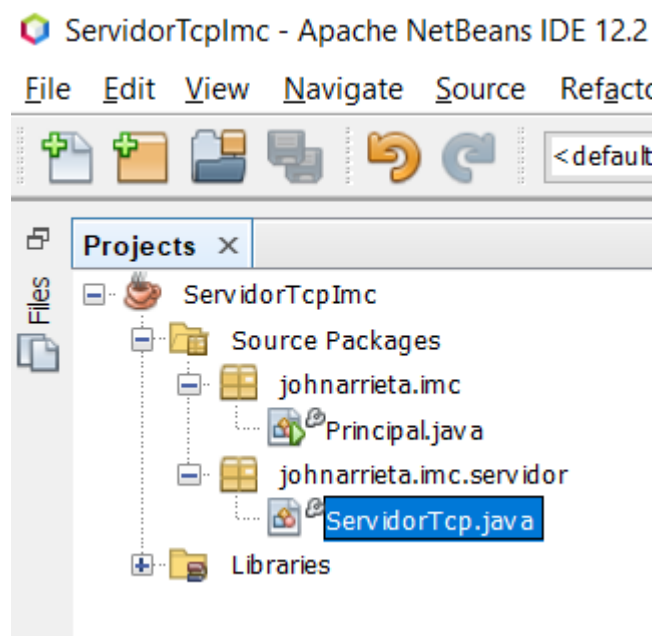
Guía básica de Desarrollo de sistemas distribuidos usando Socket TCP y Java.

Autor: John Carlos Arrieta Arrieta

Le damos el nombre y seleccionamos el paquete y le damos finish



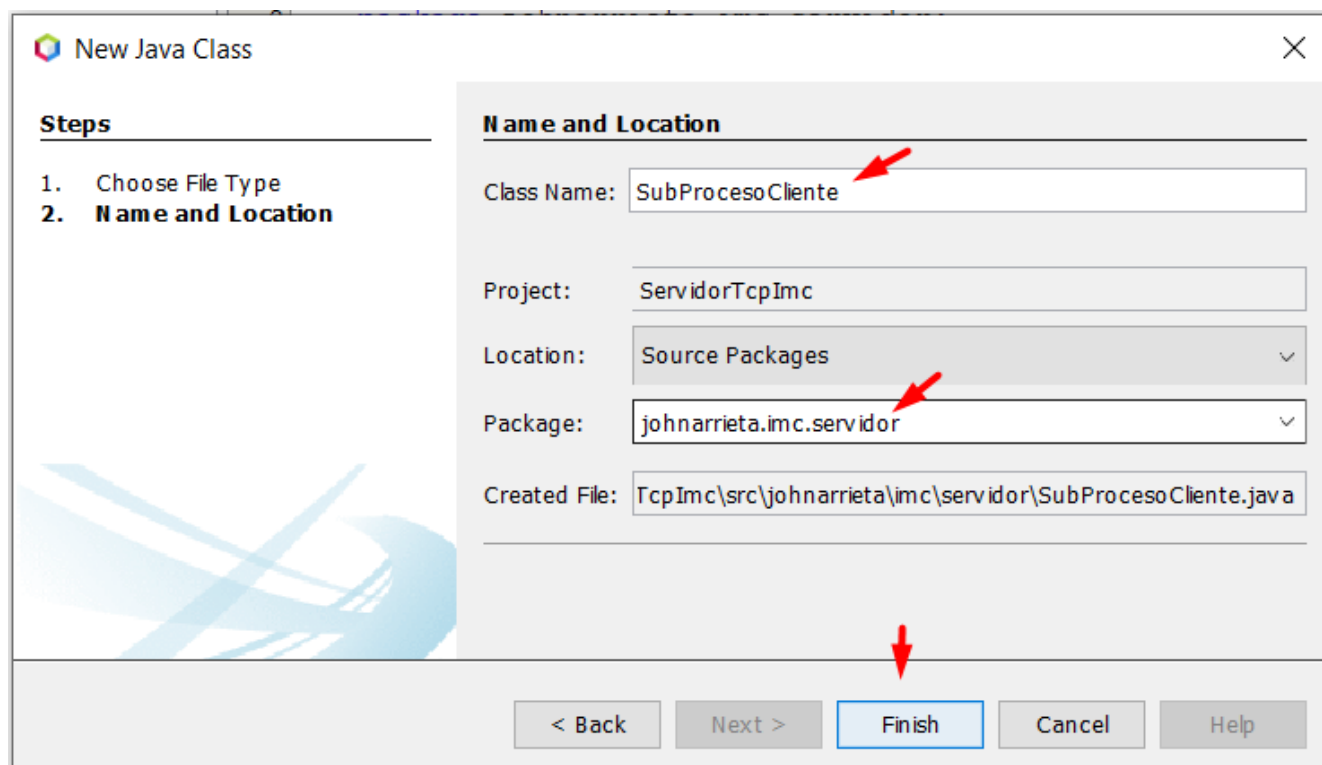
Asi quedaria



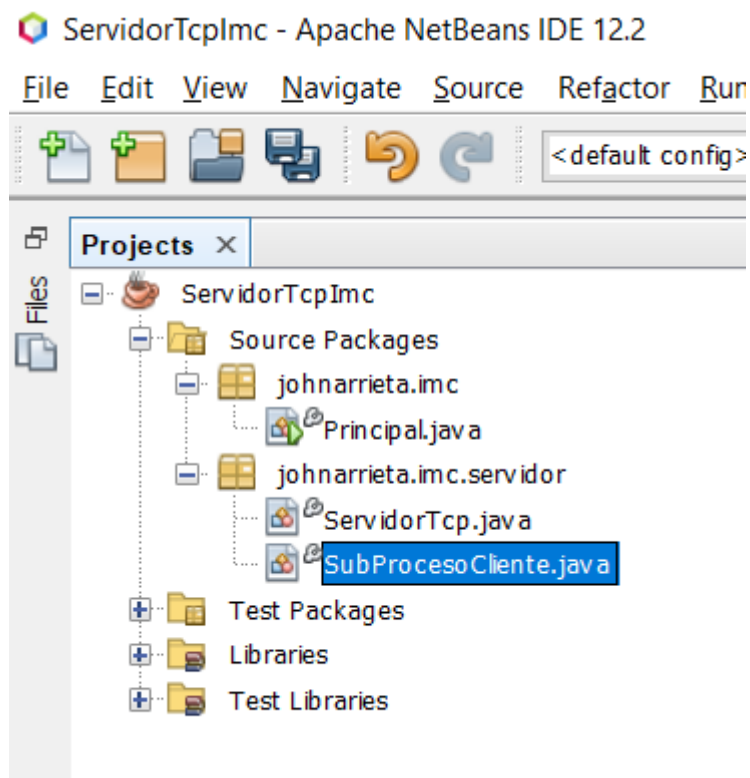
Guía básica de Desarrollo de sistemas distribuidos usando Socket TCP y Java.

Autor: John Carlos Arrieta Arrieta

Ahora creamos otra clase llamada SubProcesoCliente



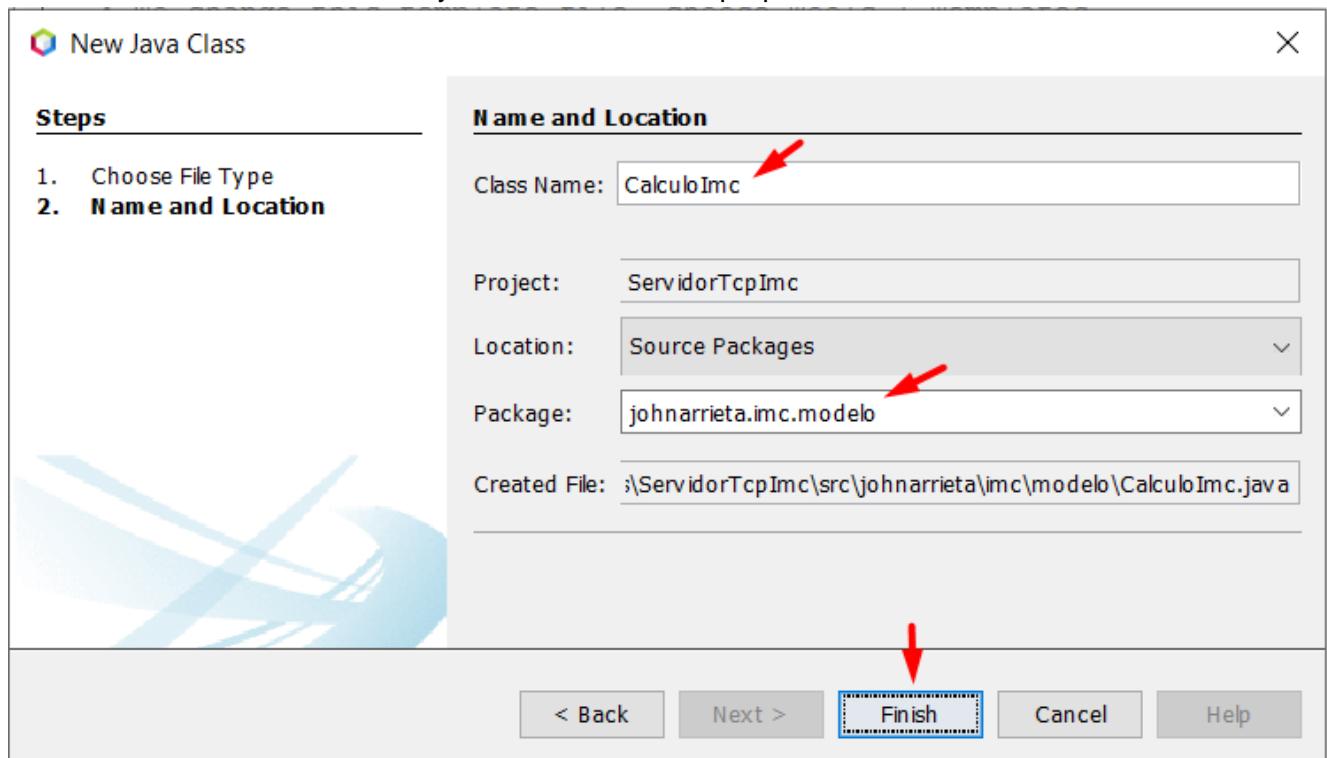
Asi queda



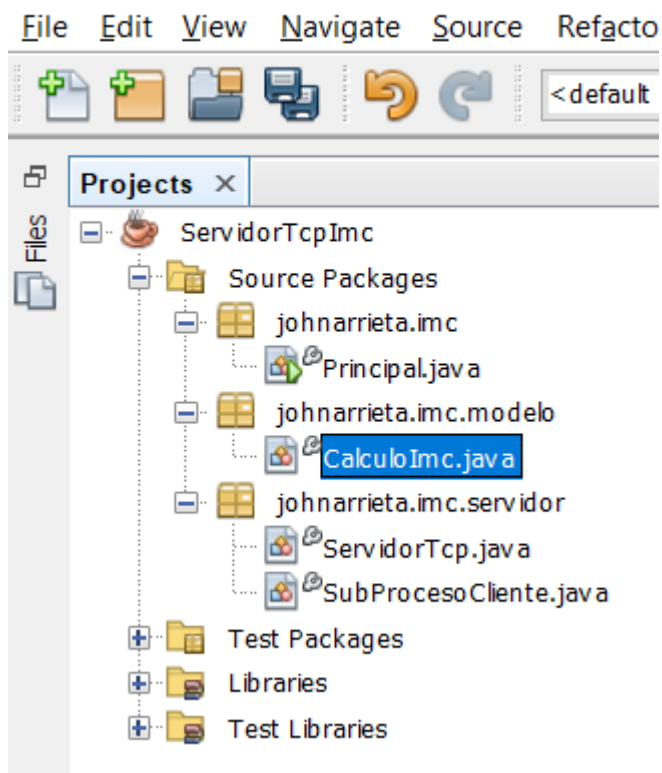
Guía básica de Desarrollo de sistemas distribuidos usando Socket TCP y Java.

Autor: John Carlos Arrieta Arrieta

Creamos la clase de CalculoImc y lo colocamos en un paquete llamado modelo




ServidorTcpImc - Apache NetBeans IDE 12.2

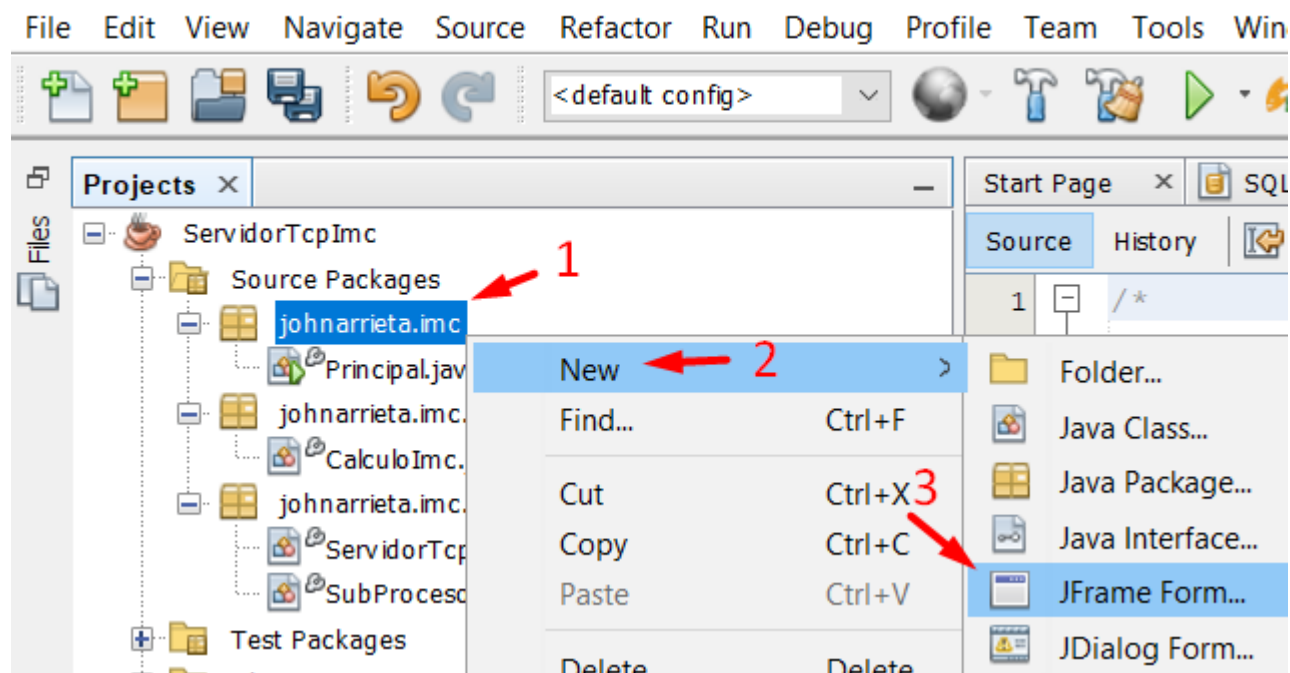


Guía básica de Desarrollo de sistemas distribuidos usando Socket TCP y Java.

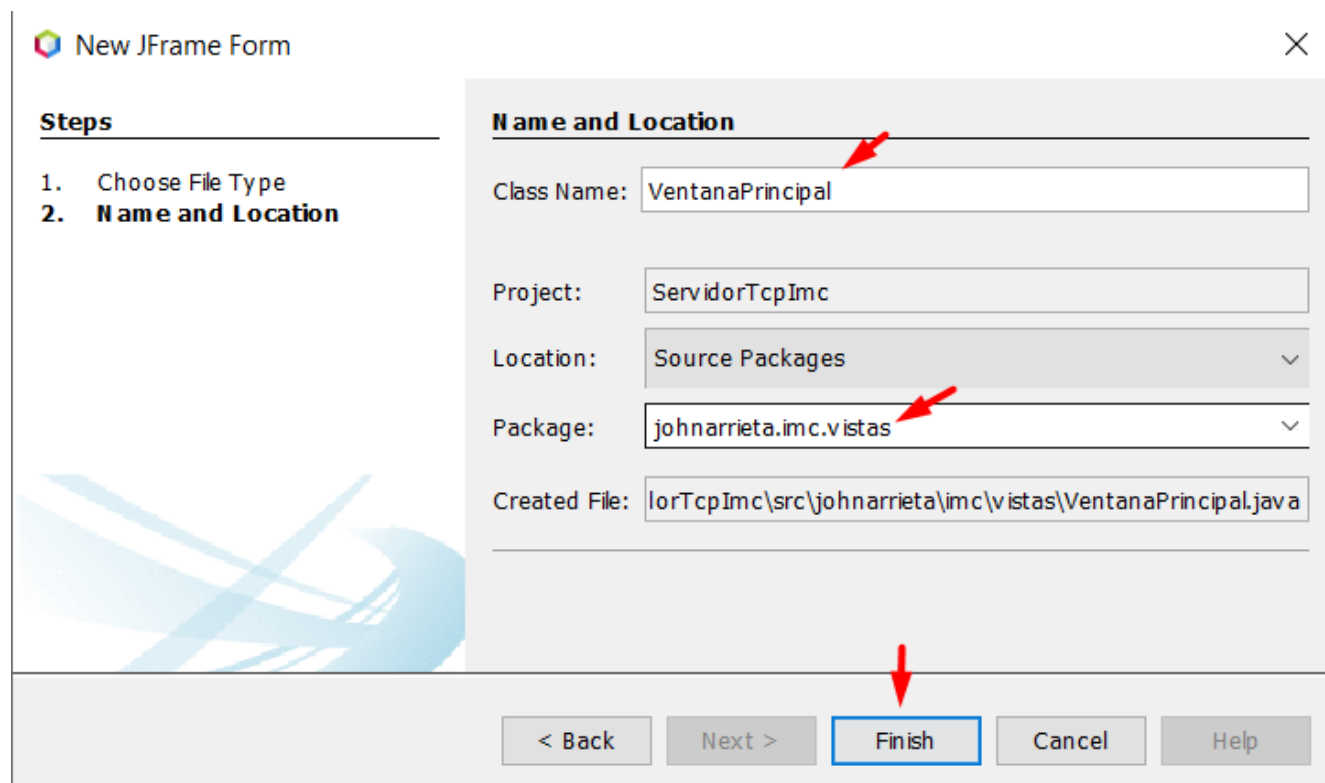
Autor: John Carlos Arrieta Arrieta

Con el JFrame crearemos nuestra vista para el servidor

 ServidorTcpImc - Apache NetBeans IDE 12.2




El nombre sera ventana principal

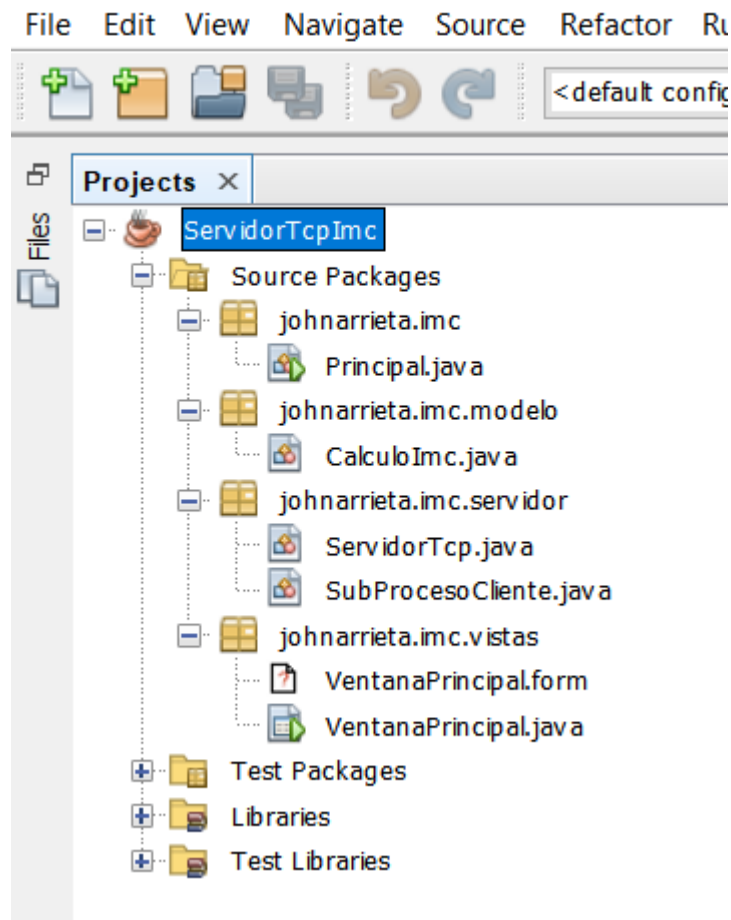


Guía básica de Desarrollo de sistemas distribuidos usando Socket TCP y Java.

Autor: John Carlos Arrieta Arrieta

Asi quedaría nuestro proyecto que es nuestro servidor

 ServidorTcpImc - Apache NetBeans IDE 12.2



Guía básica de Desarrollo de sistemas distribuidos usando Socket TCP y Java.

Autor: John Carlos Arrieta Arrieta

Aquí tenemos la clase CalculoImc donde creamos las variables, funciones necesarias para hacer el calculo Imc tomando la altura y el peso

ARCHIVO CALCULOIMC.JAVA

```
package johnarrieta.imc.modelo;

import java.io.Serializable;

/**
 * @author JOHN CARLOS ARRIETA ARRIETA
 */
public class CalculoImc implements Serializable {

    private float peso;
    private float altura;

    static class Imc {

        public float resultado;
        public String mensaje;
    }

    private Imc imc;

    public CalculoImc() { }

    public CalculoImc(float peso, float altura) {
        this.peso = peso;
        this.altura = altura;
    }

    public Imc getImc() {
        imc = new Imc();
        if (peso <= 0 || altura <= 0) {
            imc.mensaje = "ERROR: El peso y la altura deben ser mayores que 0";
            return imc;
        } else {
            imc.resultado = peso / (altura * altura);
            if (imc.resultado < 18.5) {
                imc.mensaje = "Debes consultar un Medico, tu peso es muy bajo";
            } else if (imc.resultado >= 18.5 && imc.resultado <= 24.9) {
                imc.mensaje = "Estas bien de peso";
            } else if (imc.resultado > 24.9 && imc.resultado <= 29.9) {
                imc.mensaje = "Debes bajar un poco de peso";
            } else {
                imc.mensaje = "Debes consultar un Medico, tu peso es muy alto";
            }
        }
    }
}
```

Guía básica de Desarrollo de sistemas distribuidos usando Socket TCP y Java.

Autor: John Carlos Arrieta Arrieta

```
        return imc;
    }

}

public float getPeso() {
    return peso;
}

public void setPeso(float peso) {
    this.peso = peso;
}

public float getAltura() {
    return altura;
}

public void setAltura(float altura) {
    this.altura = altura;
}
}
```

Esta Clase permite recibir un cliente en el servidor TCP, donde recibe el peso y altura y calcula el Imc usando la clase anterior y devuelve el resultado la cliente

ARCHIVO SUBPROCESOCLIENTE.JAVA

```
package johnarrieta.imc.servidor;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.Socket;
import java.text.SimpleDateFormat;
import java.util.Date;
import johnarrieta.imc.modelo.CalculoImc;
import johnarrieta.imc.vistas.VentanaPrincipal;

/**
 *
 * @author JOHN CARLOS ARRIETA ARRIETA
 */
public class SubProcesoCliente extends Thread {

    private Socket cliente;
```


Guía básica de Desarrollo de sistemas distribuidos usando Socket TCP y Java.

Autor: John Carlos Arrieta Arrieta

```
private String ip;
private VentanaPrincipal ventana;

public SubProcesoCliente(Socket cliente, VentanaPrincipal v) {
    this.cliente = cliente;
    ip = cliente.getInetAddress().getHostAddress();
    ventana = v;
}

@Override
public void run() {
    try {
        CalculoImc.Imc imc = calcularImc();
        enviarRespuesta(imc);
    } catch (Exception ex) {
        System.out.println(log()+ex.getMessage());
        ventana.getCajaLog().append(log()+ex.getMessage() + "\n");
        try {
            cliente.close();
        } catch (IOException ex1) {
            ServidorTcp.listaDeClientes.remove(ip);
        } finally {
            ServidorTcp.listaDeClientes.remove(ip);
        }
    }
}

public CalculoImc.Imc calcularImc() throws Exception {
    DataInputStream input = null;
    try {
        input = new DataInputStream(cliente.getInputStream());
        String msg = "Esperando el PESO: ";
        System.out.println(log()+msg);
        ventana.getCajaLog().append(log()+msg + "\n" + "\n");
        float peso = input.readFloat();
        msg = "PESO: " + peso;
        System.out.println(log()+msg);
        ventana.getCajaLog().append(log()+msg + "\n");
        msg = "Esperando La Altura: ";
        System.out.println(log()+msg);
        ventana.getCajaLog().append(log()+msg + "\n");
        float altura = input.readFloat();
        msg = "ALTURA: " + altura;
        System.out.println(log()+msg);
        ventana.getCajaLog().append(log()+msg + "\n");
        CalculoImc datosImc = new CalculoImc(peso, altura);
    }
}
```

Guía básica de Desarrollo de sistemas distribuidos usando Socket TCP y Java.

Autor: John Carlos Arrieta Arrieta

```
        System.out.println(log()+"IMC: " + datosImc.getImc().resultado);
        msg = "IMC: " + datosImc.getImc().resultado;
        System.out.println(log()+msg);
        ventana.getCajaLog().append(log()+msg + "\n");
        System.out.println(log()+"MENSAJE: " + datosImc.getImc().mensaje);
        msg = "MENSAJE: " + datosImc.getImc().mensaje;
        System.out.println(log()+msg);
        ventana.getCajaLog().append(log()+msg + "\n");
        return datosImc.getImc();
    } catch (IOException ex) {
        String msg = "Error al capturar datos del cliente " + ip;
        System.out.println(log()+msg);
        ventana.getCajaLog().append(log()+msg + "\n");
        throw new Exception("Error al capturar datos del cliente " + ip);
    }

}

}

public void enviarRespuesta(CalculoImc.Imc imc) {

    Thread hiloResponde = new Thread() {
        @Override
        public void run() {
            DataOutputStream output = null;
            try {
                output = new DataOutputStream(cliente.getOutputStream());
                output.writeFloat(imc.resultado);
                output.writeUTF(imc.mensaje);
                String msg = "IMC: " + imc.resultado;
                System.out.println(log()+msg);
                ventana.getCajaLog().append(log()+msg + "\n");
                msg = "MENSAJE: " + imc.mensaje;
                System.out.println(log()+msg);
                ventana.getCajaLog().append(log()+msg + "\n");
                output.flush();
                enviarRespuesta(calcularImc());
            } catch (IOException ex) {
                String msg = "Error al enviar datos al cliente " + ip;
                System.out.println(log()+msg);
                ventana.getCajaLog().append(log()+msg + "\n");
                ServidorTcp.listaDeClientes.remove(ip);
            } catch (Exception ex) {
                String msg = "Error al leer datos del cliente " + ip;
                System.out.println(log()+msg);
                ventana.getCajaLog().append(log()+msg + "\n");
            }
        }
    };
}
```

Guía básica de Desarrollo de sistemas distribuidos usando Socket TCP y Java.

Autor: John Carlos Arrieta Arrieta

```
        cliente.close();
    } catch (IOException ex1) {
        ServidorTcp.listaDeClientes.remove(ip);
    } finally {
        ServidorTcp.listaDeClientes.remove(ip);
    }
}

};
hiloResponde.start();
}

public Socket getCliente() {
    return cliente;
}

public void setCliente(Socket cliente) {
    this.cliente = cliente;
}

public String log() {
    SimpleDateFormat f = new SimpleDateFormat("dd-MM-yyyy hh:mm:ss a");
    return ip + " -> " + f.format(new Date()) + " - ";
}
}
```

En esta clase se configura el servido para escuchar en el puerto por defecto y el que se elija también guarda un registro de los usuarios que se conectan al servidor también se encarga de manejar la vista de nuestro servidor registra los logs inicia y finaliza el servidor.

ARCHIVO SERVIDORTCP.JAVA

```
package johnarrieta.imc.servidor;

import java.awt.Color;
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.HashMap;
import java.util.Map;
import java.util.function.Function;
import johnarrieta.imc.vistas.VentanaPrincipal;
```

Guía básica de Desarrollo de sistemas distribuidos usando Socket TCP y Java.

Autor: John Carlos Arrieta Arrieta

```
/**
 * @author John Carlos Arrieta Arrieta
 */
public class ServidorTcp extends Thread {

    private Boolean estado;
    public static Map<String, SubProcesoCliente> listaDeClientes;
    private Integer puerto = 9007;
    private ServerSocket servicio;
    private VentanaPrincipal ventana;

    public ServidorTcp(Integer puerto, VentanaPrincipal v) {
        if (puerto != null || puerto != 0) {
            this.puerto = puerto;
        }
        ventana = v;
        listaDeClientes = new HashMap<>();
    }

    @Override
    public void run() {
        super.run(); //To change body of generated methods, choose Tools |
Templates.
        iniciarServicio();
    }

    public void iniciarServicio() {
        try {
            servicio = new ServerSocket(puerto);
            estado = true;
            ventana.getBtnIniciar().setText("DETENER");
            ventana.getTxtEstado().setText("ONLINE");
            ventana.getTxtEstado().setForeground(Color.green);
            ventana.getBtnIniciar().setForeground(Color.RED);

            String msg = log() + "Servidor disponible en el Puerto " + puerto;
            System.out.println(msg);
            ventana.getCajaLog().append(msg + "\n");
            while (estado) {
                Socket cliente = servicio.accept();
                String ip = cliente.getInetAddress().getHostAddress();
                msg = log() + "Cliente " + ip + " conectado";
                System.out.println(msg + "\n");
                ventana.getCajaLog().append(msg + "\n");
                SubProcesoCliente atencion = new SubProcesoCliente(cliente,
ventana);
                ServidorTcp.listaDeClientes.put(ip, atencion);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private String log() {
        return "[Servidor] ";
    }
}
```

Guía básica de Desarrollo de sistemas distribuidos usando Socket TCP y Java.

Autor: John Carlos Arrieta Arrieta

```
        atencion.start();
    }
} catch (IOException ex) {
    String msg = log()+"ERROR al abrir el puerto " + puerto;
    System.out.println(msg);
    ventana.getCajaLog().append(msg + "\n");
    ventana.getBtnIniciar().setText("INICIAR");
    ventana.getTxtEstado().setText("OFF LINE");
}
}

public void detenerServicio() {
    if (estado) {
        estado = false;
        ventana.getBtnIniciar().setText("INICIAR");
        ventana.getBtnIniciar().setForeground(Color.GREEN);
        ventana.getTxtEstado().setText("OFF LINE");
        ventana.getTxtEstado().setForeground(Color.RED);
        ServidorTcp.listaDeClientes.entrySet().stream().map(new
Function<Map.Entry<String, SubProcesoCliente>, String>() {
            @Override
            public String apply(Map.Entry<String, SubProcesoCliente>
elemento) {

                String ip = elemento.getKey();
                SubProcesoCliente cliente = elemento.getValue();
                String msg = log()+"Desconectando cliente " + ip;
                System.out.println(msg);
                ventana.getCajaLog().append(msg + "\n");
                try {
                    cliente.getCliente().close();
                    cliente = null;
                    ServidorTcp.listaDeClientes.remove(elemento);
                    msg = log()+"Cliente desconectado" + ip;
                    System.out.println(msg);
                    ventana.getCajaLog().append(msg + "\n");
                } catch (IOException ex) {
                    cliente = null;
                    ServidorTcp.listaDeClientes.remove(elemento);
                    msg = log()+"Cliente desconectado" + ip;
                    System.out.println(msg);
                    ventana.getCajaLog().append(msg + "\n");
                }
                return ip;
            }
        }).forEachOrdered(ip -> {
            System.out.println("cliente " + ip + " Desconectado");
        });
    }
}
```

Guía básica de Desarrollo de sistemas distribuidos usando Socket TCP y Java.

Autor: John Carlos Arrieta Arrieta

```
        try {
            servicio.close();
        } catch (IOException ex) {
            System.out.println("ERRRO no se puede cerrar el Puerto " +
puerto);

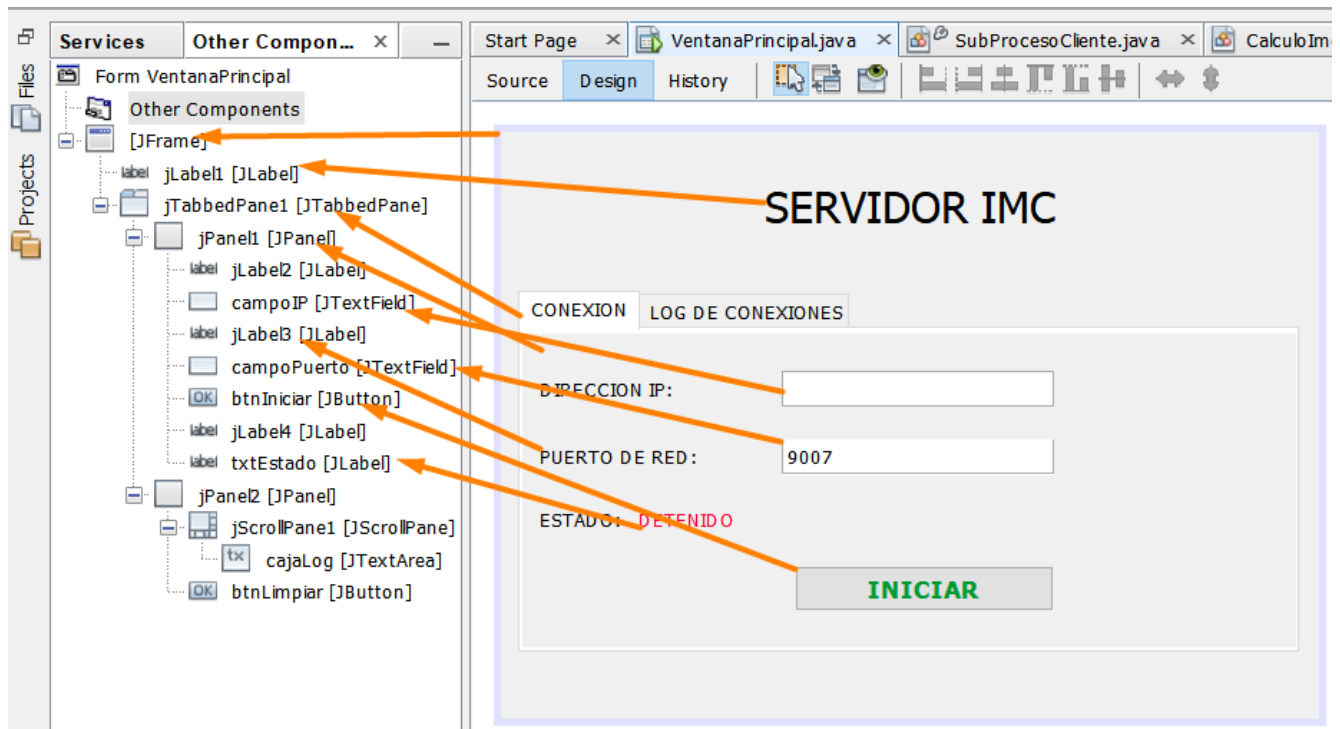
            String msg = log()+"ERRRO no se puede cerrar el Puerto " +
puerto;

            System.out.println(msg);
            ventana.getCajaLog().append(msg + "\n");
        }
    }
}

public String log() {
    SimpleDateFormat f = new SimpleDateFormat("dd-MM-yyyy hh:mm:ss a");
    return f.format(new Date()) + " - ";
}
}
```

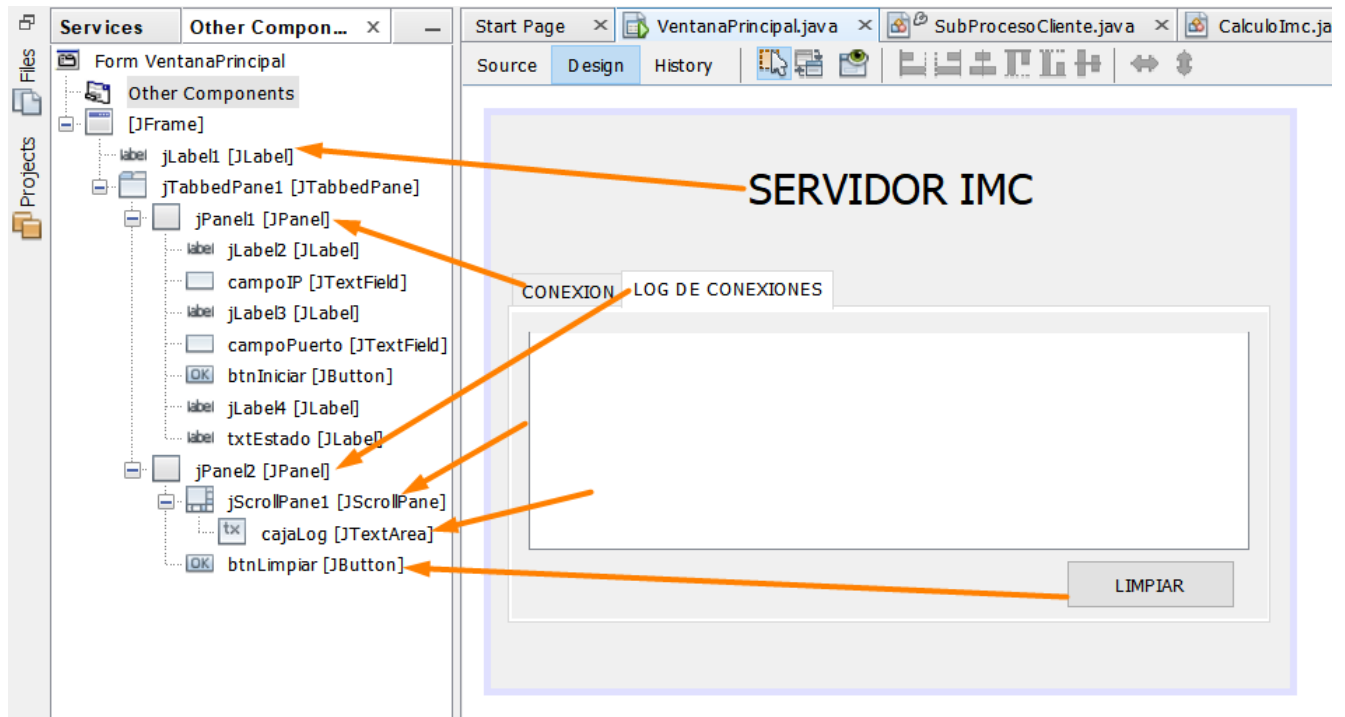
Esta es la vista del servidor donde se inicia el servidor y muestra los logs de conexión

ARCHIVO VENTANAPRINCIPAL.JAVA



Guía básica de Desarrollo de sistemas distribuidos usando Socket TCP y Java.

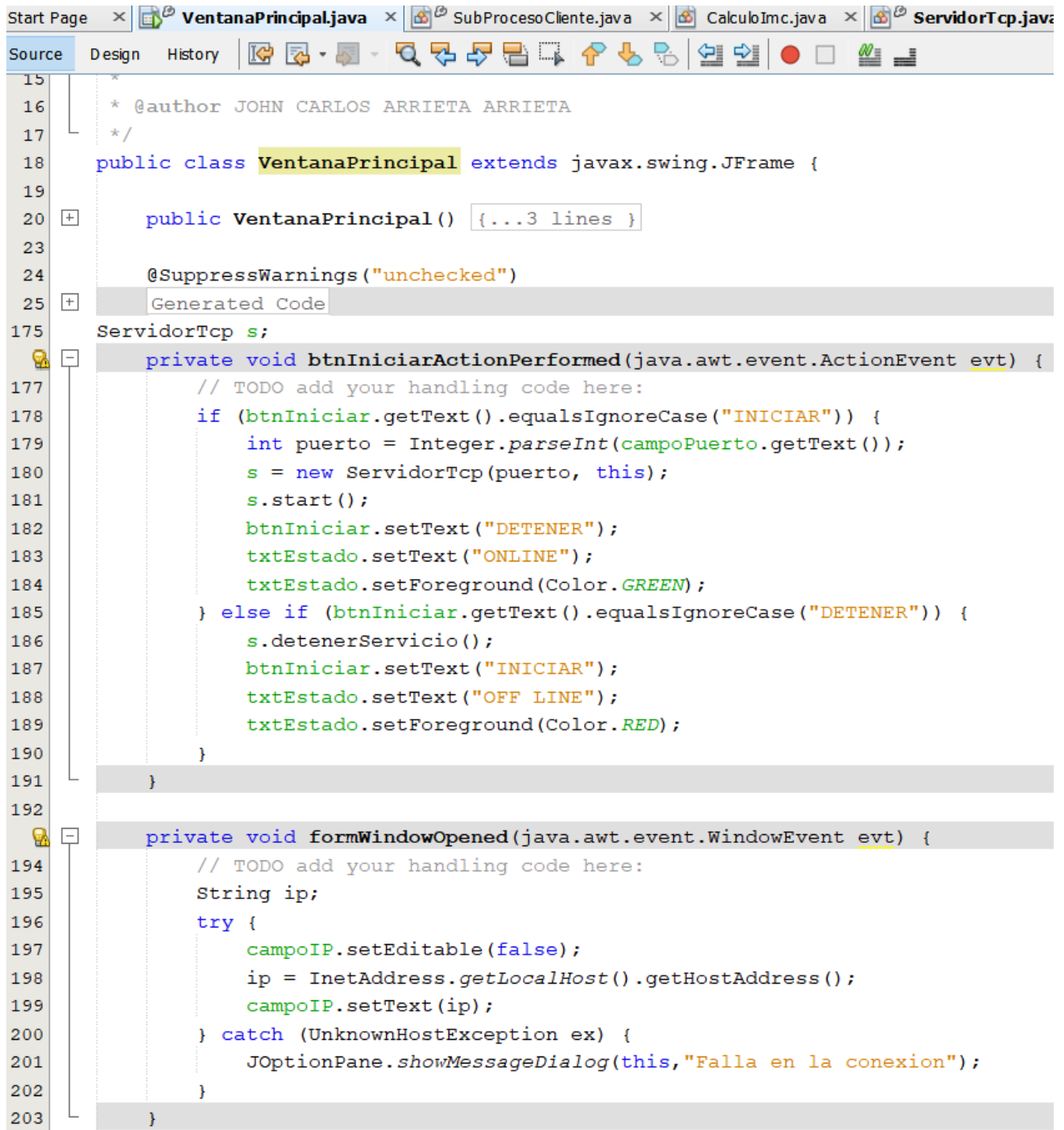
Autor: John Carlos Arrieta Arrieta



Guía básica de Desarrollo de sistemas distribuidos usando Socket TCP y Java.

Autor: John Carlos Arrieta Arrieta

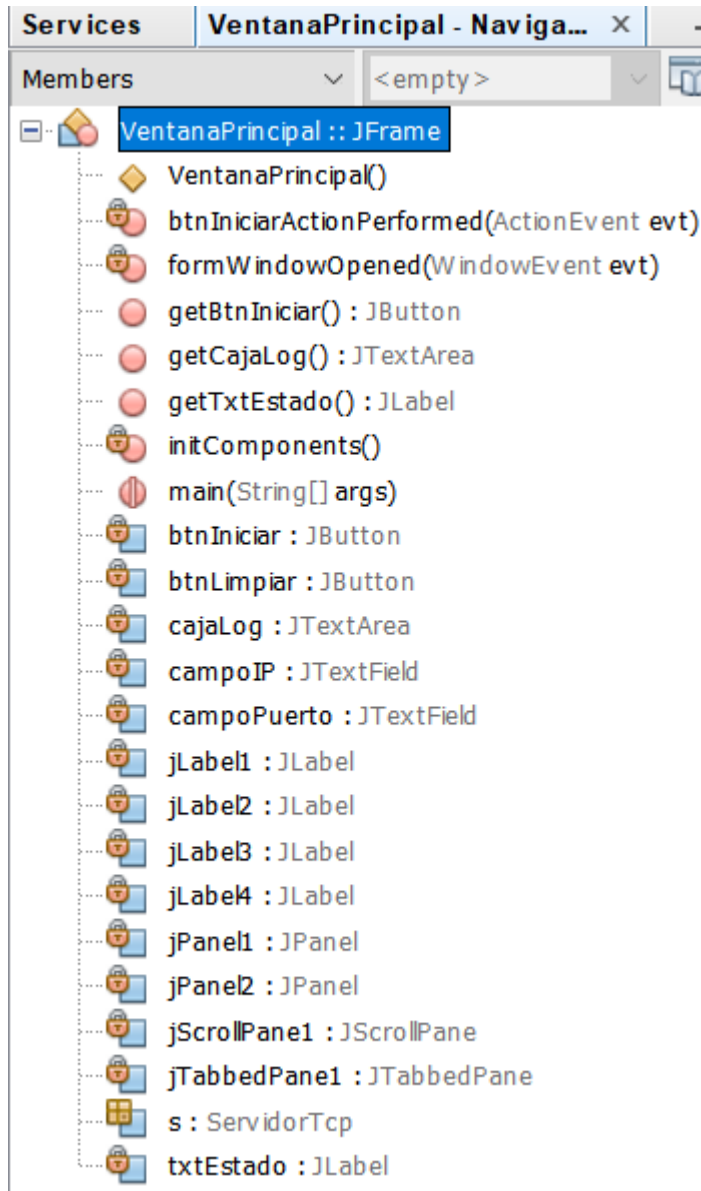
Esta es la configuración de las acciones de cada boton el de iniciar el servidor y el detener el servidor



```
15  *
16  * @author JOHN CARLOS ARRIETA ARRIETA
17  */
18  public class VentanaPrincipal extends javax.swing.JFrame {
19
20      public VentanaPrincipal() {...3 lines }
21
22      @SuppressWarnings("unchecked")
23
24      Generated Code
25
26  ServidorTcp s;
27
28  private void btnIniciarActionPerformed(java.awt.event.ActionEvent evt) {
29      // TODO add your handling code here:
30      if (btnIniciar.getText().equalsIgnoreCase("INICIAR")) {
31          int puerto = Integer.parseInt(campoPuerto.getText());
32          s = new ServidorTcp(puerto, this);
33          s.start();
34          btnIniciar.setText("DETENER");
35          txtEstado.setText("ONLINE");
36          txtEstado.setForeground(Color.GREEN);
37      } else if (btnIniciar.getText().equalsIgnoreCase("DETENER")) {
38          s.detenerServicio();
39          btnIniciar.setText("INICIAR");
40          txtEstado.setText("OFF LINE");
41          txtEstado.setForeground(Color.RED);
42      }
43  }
44
45  private void formWindowOpened(java.awt.event.WindowEvent evt) {
46      // TODO add your handling code here:
47      String ip;
48      try {
49          campoIP.setEditable(false);
50          ip = InetAddress.getLocalHost().getHostAddress();
51          campoIP.setText(ip);
52      } catch (UnknownHostException ex) {
53          JOptionPane.showMessageDialog(this, "Falla en la conexion");
54      }
55  }
```


Guía básica de Desarrollo de sistemas distribuidos usando Socket TCP y Java.

Autor: John Carlos Arrieta Arrieta



```
package johnarrieta.imc.vistas;

import java.awt.Color;
import java.net.InetAddress;
import java.net.UnknownHostException;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JTextArea;
import johnarrieta.imc.servidor.ServidorTcp;

/**
 *
 * @author JOHN CARLOS ARRIETA ARRIETA
 */
```

Guía básica de Desarrollo de sistemas distribuidos usando Socket TCP y Java.

Autor: John Carlos Arrieta Arrieta

```
public class VentanaPrincipal extends javax.swing.JFrame {

    public VentanaPrincipal() {
        initComponents();
    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jLabel1 = new javax.swing.JLabel();
        jTabbedPane1 = new javax.swing.JTabbedPane();
        jPanel1 = new javax.swing.JPanel();
        jLabel2 = new javax.swing.JLabel();
        campoIP = new javax.swing.JTextField();
        jLabel3 = new javax.swing.JLabel();
        campoPuerto = new javax.swing.JTextField();
        btnIniciar = new javax.swing.JButton();
        jLabel4 = new javax.swing.JLabel();
        txtEstado = new javax.swing.JLabel();
        jPanel2 = new javax.swing.JPanel();
        jScrollPane1 = new javax.swing.JScrollPane();
        cajaLog = new javax.swing.JTextArea();
        btnLimpiar = new javax.swing.JButton();

        setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
        addWindowListener(new java.awt.event.WindowAdapter() {
            public void windowOpened(java.awt.event.WindowEvent evt) {
                formWindowOpened(evt);
            }
        });

        jLabel1.setFont(new java.awt.Font("Tahoma", 0, 24)); // NOI18N
        jLabel1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
        jLabel1.setText("SERVIDOR IMC");

        jLabel2.setText("DIRECCION IP: ");

        jLabel3.setText("PUERTO DE RED:");

        campoPuerto.setText("9007");

        btnIniciar.setFont(new java.awt.Font("Tahoma", 1, 14)); // NOI18N
        btnIniciar.setForeground(new java.awt.Color(0, 153, 51));
        btnIniciar.setText("INICIAR");
        btnIniciar.addActionListener(new java.awt.event.ActionListener() {
            public void actionPerformed(java.awt.event.ActionEvent evt) {

```

Guía básica de Desarrollo de sistemas distribuidos usando Socket TCP y Java.

Autor: John Carlos Arrieta Arrieta

```
        btnIniciarActionPerformed(evt);
    }
});

jLabel4.setText("ESTADO: ");

txtEstado.setForeground(new java.awt.Color(255, 0, 51));
txtEstado.setText("DETENIDO");

javax.swing.GroupLayout jPanel1Layout = new
javax.swing.GroupLayout(jPanel1);
jPanel1.setLayout(jPanel1Layout);
jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(jPanel1Layout.createSequentialGroup()
        .addGap(10, 10, 10)
        .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addComponent(jLabel2,
javax.swing.GroupLayout.PREFERRED_SIZE, 131,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addComponent(campoIP,
javax.swing.GroupLayout.PREFERRED_SIZE, 152,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                    .addComponent(jLabel3,
javax.swing.GroupLayout.PREFERRED_SIZE, 131,
javax.swing.GroupLayout.PREFERRED_SIZE)
                    .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                        .addComponent(campoPuerto)
                        .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                            .addComponent(txtEstado,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
                            .addComponent(btnIniciar,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.PREFERRED_SIZE, 145,
javax.swing.GroupLayout.PREFERRED_SIZE))
                        .addGap(10, 10, 10)
                    )
                )
            )
        )
    )
);
```

Guía básica de Desarrollo de sistemas distribuidos usando Socket TCP y Java.

Autor: John Carlos Arrieta Arrieta

```
        .addContainerGap(135, Short.MAX_VALUE))
    );
    jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(jPanel1Layout.createSequentialGroup()
            .addGap(21, 21, 21)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BA
SELINE)
            .addComponent(jLabel12)
            .addComponent(campoIP,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(18, 18, 18)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BA
SELINE)
            .addComponent(jLabel13)
            .addComponent(campoPuerto,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(18, 18, 18)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BA
SELINE)
            .addComponent(jLabel14)
            .addComponent(txtEstado))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 19,
Short.MAX_VALUE)
            .addComponent(btnIniciar)
            .addGap(20, 20, 20))
    );

    jTabbedPane1.addTab("CONEXION", jPanel1);

    cajaLog.setEditable(false);
    cajaLog.setColumns(20);
    cajaLog.setRows(5);
    jScrollPane1.setViewportView(cajaLog);

    btnLimpiar.setText("LIMPIAR");

    javax.swing.GroupLayout jPanel2Layout = new
javax.swing.GroupLayout(jPanel2);
    jPanel2.setLayout(jPanel2Layout);
```

Guía básica de Desarrollo de sistemas distribuidos usando Socket TCP y Java.

Autor: John Carlos Arrieta Arrieta

```
jPanel2Layout.setHorizontalGroup(

jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
jPanel2Layout.createSequentialGroup())

.addGroup(jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
    .addGroup(jPanel2Layout.createSequentialGroup()
        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
        .addComponent(btnLimpiar,
javax.swing.GroupLayout.PREFERRED_SIZE, 97,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addGap(8, 8, 8))
        .addGroup(jPanel2Layout.createSequentialGroup()
            .addContainerGap()
            .addComponent(jScrollPane1,
javax.swing.GroupLayout.DEFAULT_SIZE, 412, Short.MAX_VALUE)))
        .addContainerGap())
    );
jPanel2Layout.setVerticalGroup(

jPanel2Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(jPanel2Layout.createSequentialGroup()
        .addContainerGap()
        .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 125,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
    .addComponent(btnLimpiar, javax.swing.GroupLayout.DEFAULT_SIZE,
28, Short.MAX_VALUE)
    .addGap(5, 5, 5))
    );

jTabbedPane1.addTab("LOG DE CONEXIONES", jPanel2);

javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
        .addContainerGap()
```

Guía básica de Desarrollo de sistemas distribuidos usando Socket TCP y Java.

Autor: John Carlos Arrieta Arrieta

```
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addComponent(jLabel1,
javax.swing.GroupLayout.PREFERRED_SIZE, 437,
javax.swing.GroupLayout.PREFERRED_SIZE)
    .addComponent(jTabbedPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
    .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
);
layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
        .addGap(27, 27, 27)
        .addComponent(jLabel1)
        .addGap(32, 32, 32)
        .addComponent(jTabbedPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 203,
javax.swing.GroupLayout.PREFERRED_SIZE)
        .addContainerGap(37, Short.MAX_VALUE))
    );

pack();
} // </editor-fold>
ServidorTcp s;
private void btnIniciarActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    if (btnIniciar.getText().equalsIgnoreCase("INICIAR")) {
        int puerto = Integer.parseInt(campoPuerto.getText());
        s = new ServidorTcp(puerto, this);
        s.start();
        btnIniciar.setText("DETENER");
        txtEstado.setText("ONLINE");
        txtEstado.setForeground(Color.GREEN);
    } else if (btnIniciar.getText().equalsIgnoreCase("DETENER")) {
        s.detenerServicio();
        btnIniciar.setText("INICIAR");
        txtEstado.setText("OFF LINE");
        txtEstado.setForeground(Color.RED);
    }
}

private void formWindowOpened(java.awt.event.WindowEvent evt) {
    // TODO add your handling code here:
    String ip;
    try {
```

Guía básica de Desarrollo de sistemas distribuidos usando Socket TCP y Java.

Autor: John Carlos Arrieta Arrieta

```
        campoIP.setEditable(false);
        ip = InetAddress.getLocalHost().getHostAddress();
        campoIP.setText(ip);
    } catch (UnknownHostException ex) {
        JOptionPane.showMessageDialog(this,"Falla en la conexion");
    }
}

public JLabel getTxtEstado() {
    return txtEstado;
}

public JTextArea getCajaLog() {
    return cajaLog;
}

public JButton getBtnIniciar() {
    return btnIniciar;
}

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus Look and feel */
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting
code (optional) ">
    /* If Nimbus (introduced in Java SE 6) is not available, stay with the
default look and feel.
    * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
    */
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    } catch (ClassNotFoundException ex) {

        java.util.logging.Logger.getLogger(VentanaPrincipal.class.getName()).log(java.ut
il.logging.Level.SEVERE, null, ex);
    } catch (InstantiationException ex) {

        java.util.logging.Logger.getLogger(VentanaPrincipal.class.getName()).log(java.ut
```

Guía básica de Desarrollo de sistemas distribuidos usando Socket TCP y Java.

Autor: John Carlos Arrieta Arrieta

```
il.logging.Level.SEVERE, null, ex);
    } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(VentanaPrincipal.class.getName()).log(java.ut
il.logging.Level.SEVERE, null, ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(VentanaPrincipal.class.getName()).log(java.ut
il.logging.Level.SEVERE, null, ex);
    }
    //</editor-fold>

    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable() {
        public void run() {
            new VentanaPrincipal().setVisible(true);
        }
    });
}

// Variables declaration - do not modify
private javax.swing.JButton btnIniciar;
private javax.swing.JButton btnLimpiar;
private javax.swing.JTextArea cajaLog;
private javax.swing.JTextField campoIP;
private javax.swing.JTextField campoPuerto;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;
private javax.swing.JPanel jPanel1;
private javax.swing.JPanel jPanel2;
private javax.swing.JScrollPane jScrollPane1;
private javax.swing.JTabbedPane jTabbedPane1;
private javax.swing.JLabel txtEstado;
    // End of variables declaration
}
```

Este es el archivo principal donde se ejecuta el servidor para mostrar la ventana principal
ARCHIVO PRINCIPAL.JAVA

```
package johnarrieta.imc;

import johnarrieta.imc.vistas.VentanaPrincipal;

/**
 * @author JOHN CARLOS ARRIETA ARRIETA
 */
```

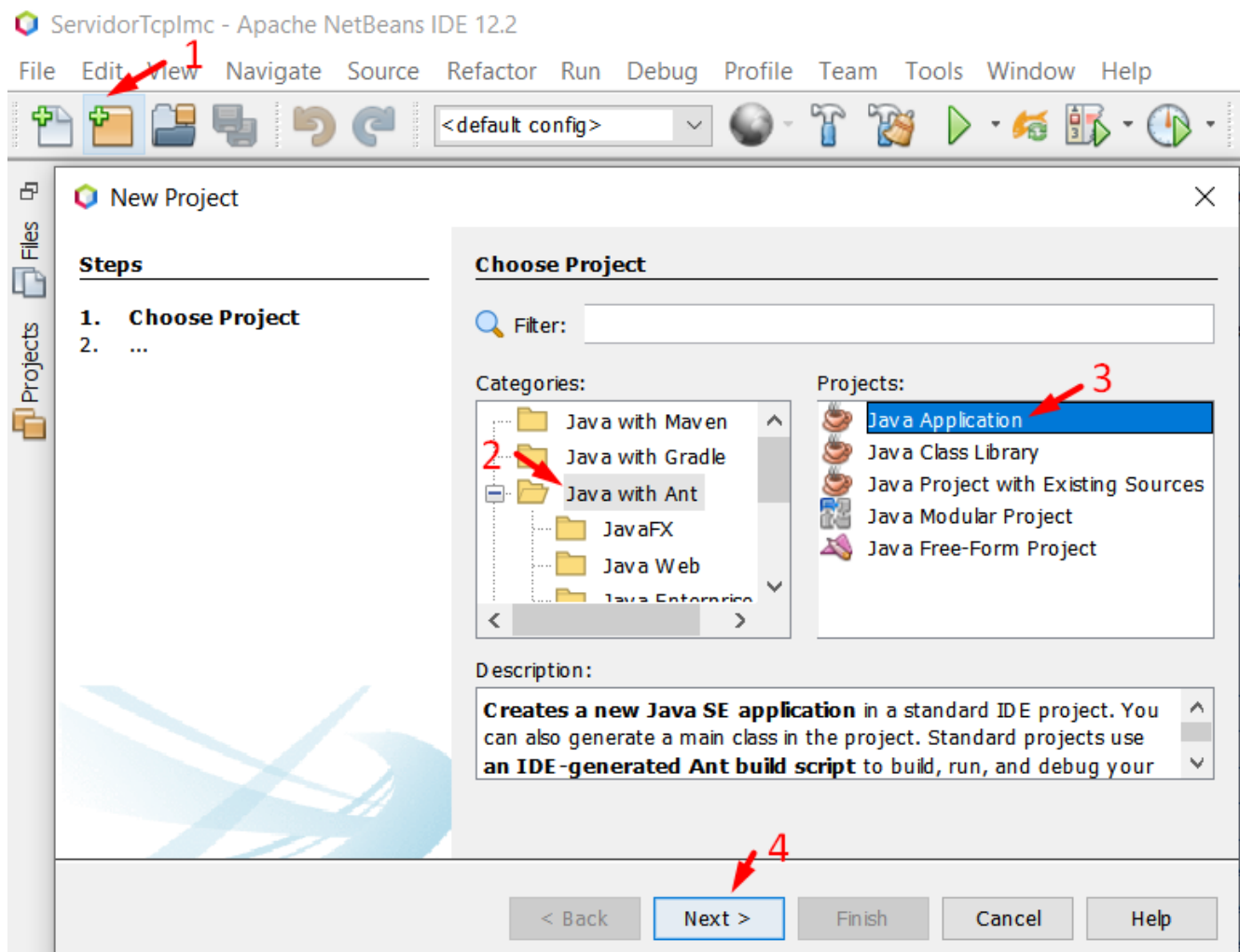

Guía básica de Desarrollo de sistemas distribuidos usando Socket TCP y Java.

Autor: John Carlos Arrieta Arrieta

```
public class Principal {  
  
    /**  
     * @param args the command line arguments  
     */  
    public static void main(String[] args) {  
        VentanaPrincipal v = new VentanaPrincipal();  
        v.setLocationRelativeTo(null);  
        v.setVisible(true);  
    }  
}
```

En este apartado estaremos creando la aplicación cliente de la misma manera del servidor

LA APLICACION CLIENTE



Guía básica de Desarrollo de sistemas distribuidos usando Socket TCP y Java.

Autor: John Carlos Arrieta Arrieta

New Java Application

Steps

1. Choose Project
2. **Name and Location**

Name and Location

Project Name:

Project Location:

Project Folder:

☐ Use Dedicated Folder for Storing Libraries

Libraries Folder:

Different users and projects can share the same compilation libraries (see Help for details).

☒ Create Main Class

New Java Application

Steps

1. Choose Project
2. **Name and Location**

Name and Location

Project Name:

Project Location:

Project Folder:

☐ Use Dedicated Folder for Storing Libraries

Libraries Folder:

Different users and projects can share the same compilation libraries (see Help for details).

☒ Create Main Class

Guía básica de Desarrollo de sistemas distribuidos usando Socket TCP y Java.

Autor: John Carlos Arrieta Arrieta

New Java Application

Steps

1. Choose Project
2. **Name and Location**

Name and Location

Project Name: 1

Project Location: Browse...

Project Folder:

☐ Use Dedicated Folder for Storing Libraries

Libraries Folder: Browse...

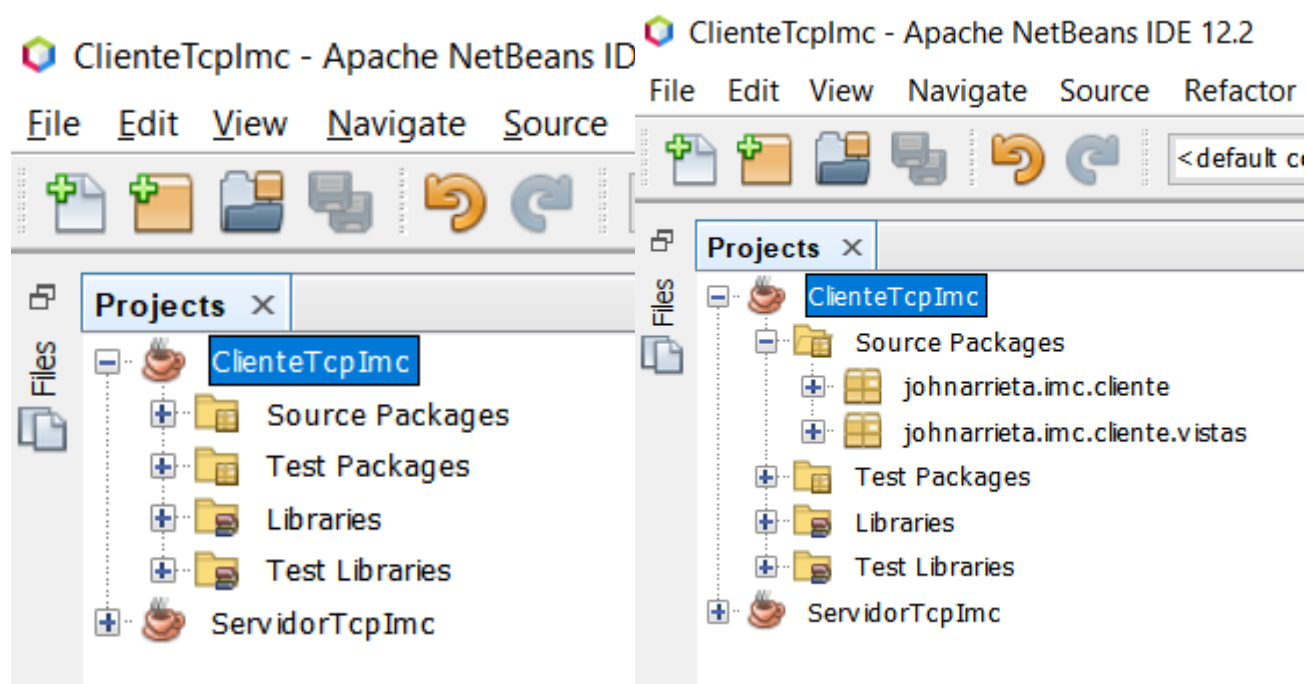
Different users and projects can share the same compilation libraries (see Help for details).

☒ Create Main Class 3

2

4

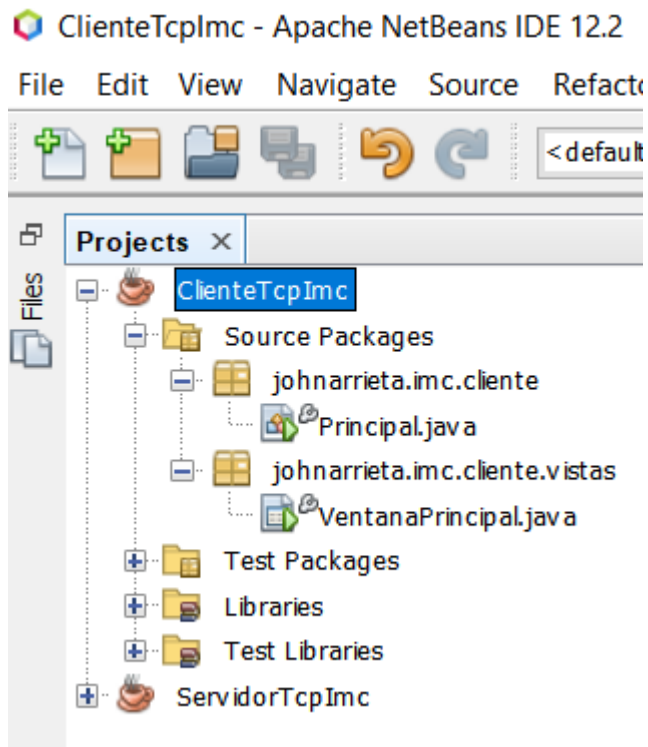
< Back Next > **Finish** Cancel Help



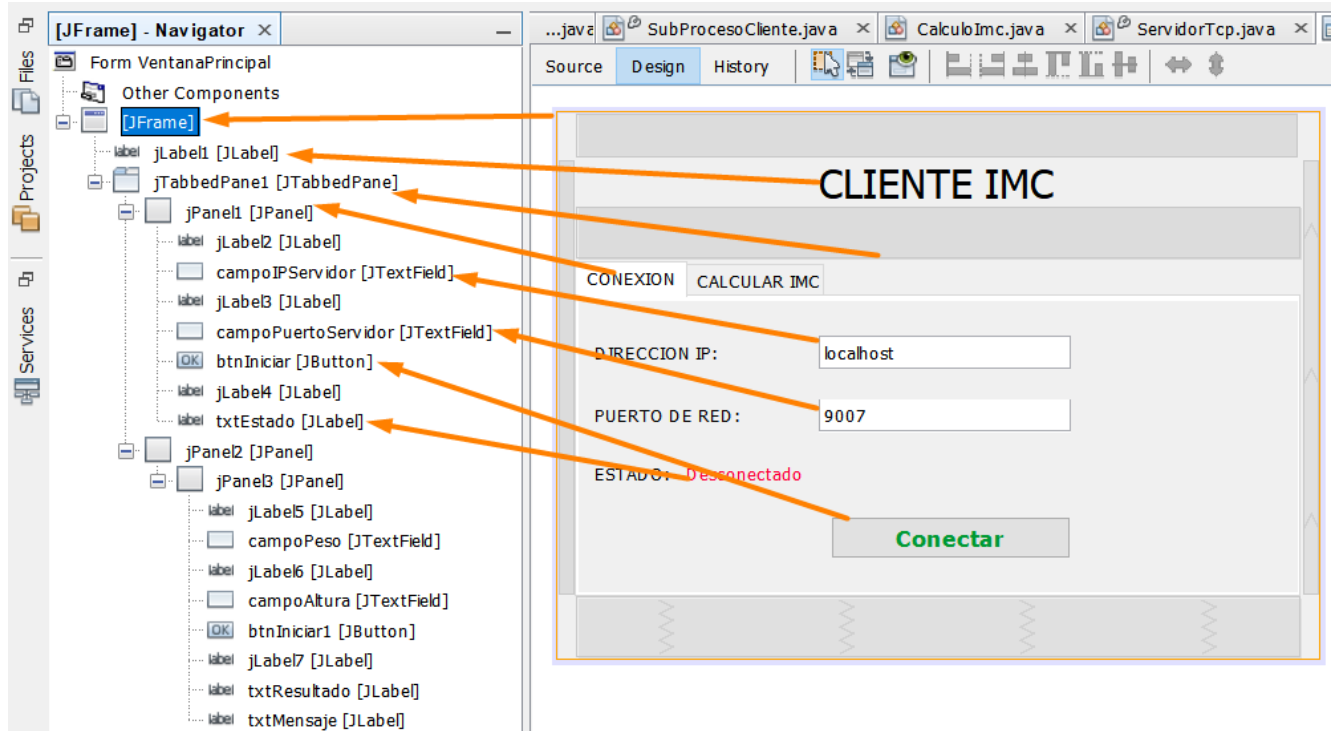
Guía básica de Desarrollo de sistemas distribuidos usando Socket TCP y Java.

Autor: John Carlos Arrieta Arrieta

El creamos un paquete donde estara el archivo principal y otra donde estara la vista del cliente donde se conectara con el servidor y permitira realizar el calculo IMC

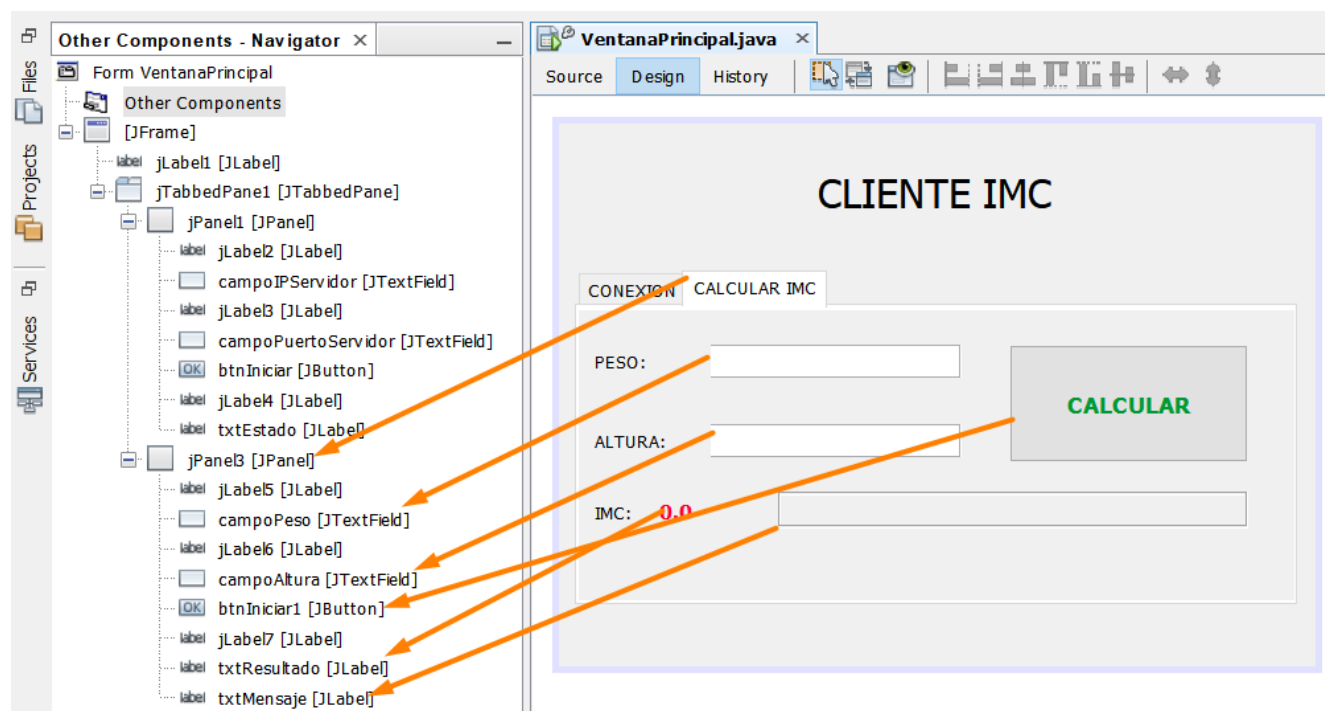


Esta esta vista que muestra la manera de conectarse al servidor



Guía básica de Desarrollo de sistemas distribuidos usando Socket TCP y Java.

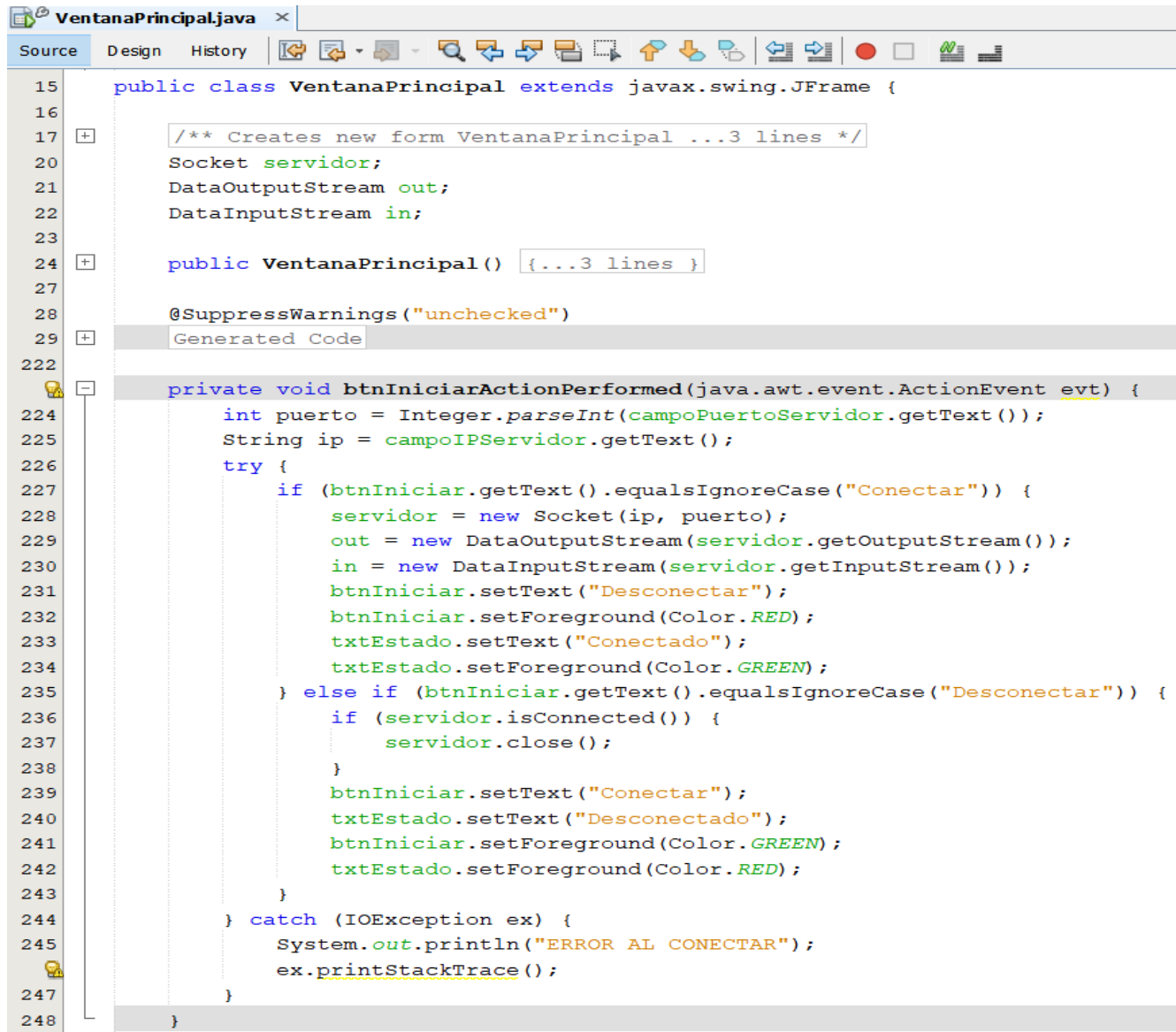
Autor: John Carlos Arrieta Arrieta



Guía básica de Desarrollo de sistemas distribuidos usando Socket TCP y Java.

Autor: John Carlos Arrieta Arrieta

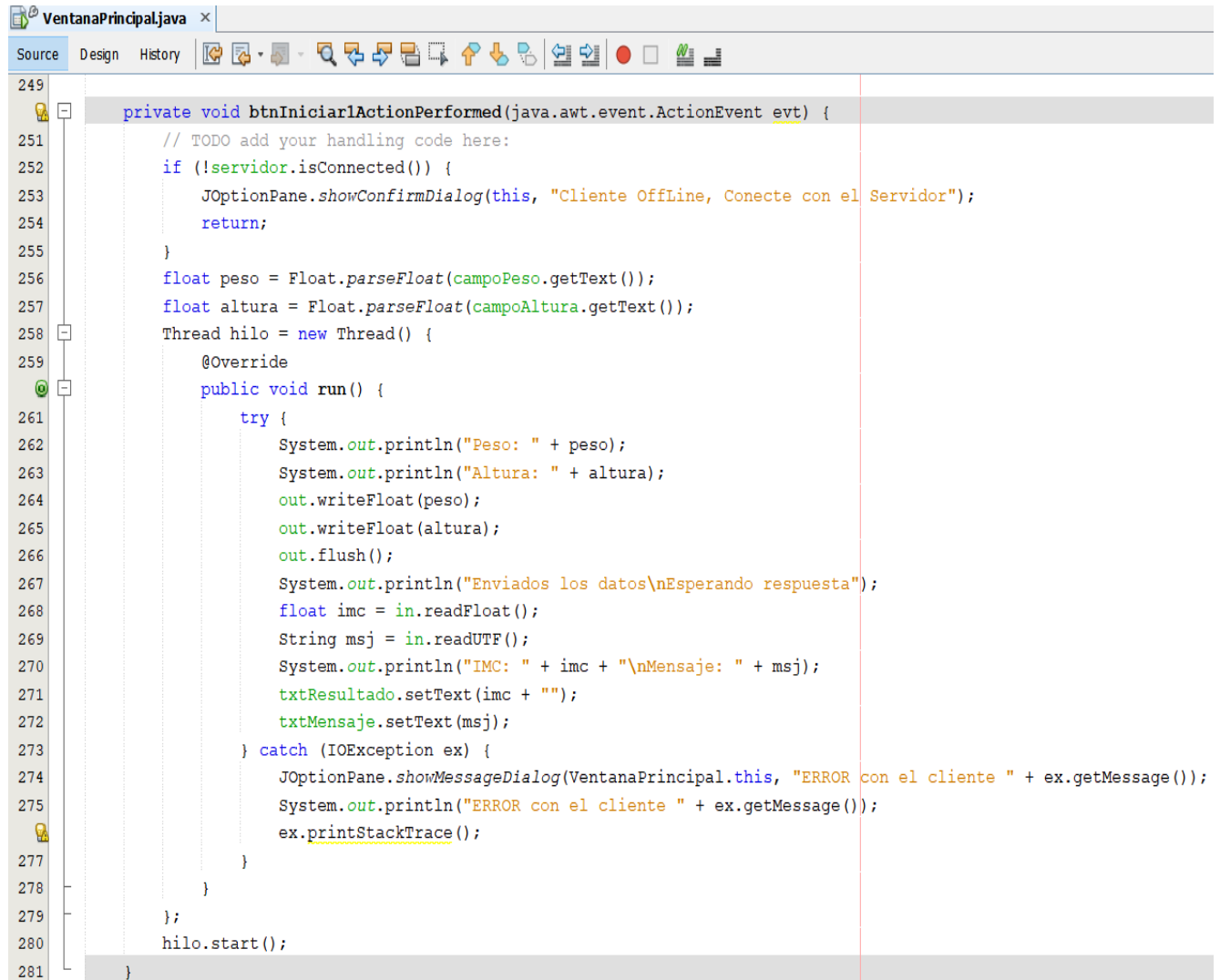
Aquí mostramos las acciones de los dos botones al presionar el boton conectar con el servidor y calcular el imc



```
15 public class VentanaPrincipal extends javax.swing.JFrame {
16
17     /** Creates new form VentanaPrincipal ...3 lines */
20     Socket servidor;
21     DataOutputStream out;
22     DataInputStream in;
23
24     public VentanaPrincipal() { ...3 lines }
27
28     @SuppressWarnings("unchecked")
29     Generated Code
222
223     private void btnIniciarActionPerformed(java.awt.event.ActionEvent evt) {
224         int puerto = Integer.parseInt(campoPuertoServidor.getText());
225         String ip = campoIPServidor.getText();
226         try {
227             if (btnIniciar.getText().equalsIgnoreCase("Conectar")) {
228                 servidor = new Socket(ip, puerto);
229                 out = new DataOutputStream(servidor.getOutputStream());
230                 in = new DataInputStream(servidor.getInputStream());
231                 btnIniciar.setText("Desconectar");
232                 btnIniciar.setForeground(Color.RED);
233                 txtEstado.setText("Conectado");
234                 txtEstado.setForeground(Color.GREEN);
235             } else if (btnIniciar.getText().equalsIgnoreCase("Desconectar")) {
236                 if (servidor.isConnected()) {
237                     servidor.close();
238                 }
239                 btnIniciar.setText("Conectar");
240                 txtEstado.setText("Desconectado");
241                 btnIniciar.setForeground(Color.GREEN);
242                 txtEstado.setForeground(Color.RED);
243             }
244         } catch (IOException ex) {
245             System.out.println("ERROR AL CONECTAR");
246             ex.printStackTrace();
247         }
248     }
```

Guía básica de Desarrollo de sistemas distribuidos usando Socket TCP y Java.

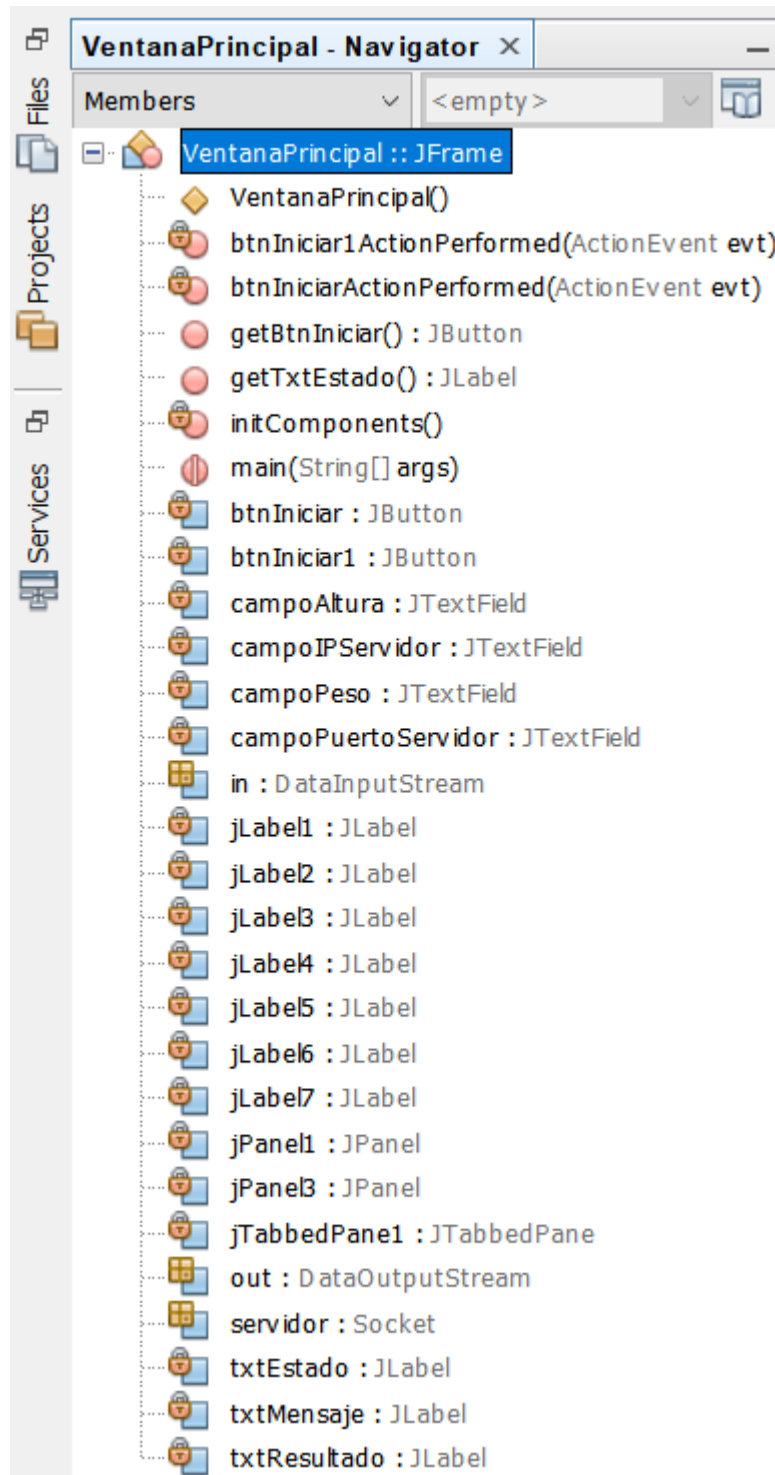
Autor: John Carlos Arrieta Arrieta



```
249 private void btnIniciarActionPerformed(java.awt.event.ActionEvent evt) {
251     // TODO add your handling code here:
252     if (!servidor.isConnected()) {
253         JOptionPane.showConfirmDialog(this, "Cliente OffLine, Conecte con el Servidor");
254         return;
255     }
256     float peso = Float.parseFloat(campoPeso.getText());
257     float altura = Float.parseFloat(campoAltura.getText());
258     Thread hilo = new Thread() {
259         @Override
260         public void run() {
261             try {
262                 System.out.println("Peso: " + peso);
263                 System.out.println("Altura: " + altura);
264                 out.writeFloat(peso);
265                 out.writeFloat(altura);
266                 out.flush();
267                 System.out.println("Enviados los datos\nEsperando respuesta");
268                 float imc = in.readFloat();
269                 String msj = in.readUTF();
270                 System.out.println("IMC: " + imc + "\nMensaje: " + msj);
271                 txtResultado.setText(imc + "");
272                 txtMensaje.setText(msj);
273             } catch (IOException ex) {
274                 JOptionPane.showMessageDialog(VentanaPrincipal.this, "ERROR con el cliente " + ex.getMessage());
275                 System.out.println("ERROR con el cliente " + ex.getMessage());
276                 ex.printStackTrace();
277             }
278         }
279     };
280     hilo.start();
281 }
```

Guía básica de Desarrollo de sistemas distribuidos usando Socket TCP y Java.

Autor: John Carlos Arrieta Arrieta



Guía básica de Desarrollo de sistemas distribuidos usando Socket TCP y Java.

Autor: John Carlos Arrieta Arrieta

Aquí esta toda la informacion que tiene este JFrame para ejecutar el cliente

```
package johnarrieta.imc.cliente.vistas;

import java.awt.Color;
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.Socket;
import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JOptionPane;

/**
 * @author JOHN CARLOS ARRIETA ARRIETA
 */
public class VentanaPrincipal extends javax.swing.JFrame {

    /**
     * Creates new form VentanaPrincipal
     */
    Socket servidor;
    DataOutputStream out;
    DataInputStream in;

    public VentanaPrincipal() {
        initComponents();
    }

    @SuppressWarnings("unchecked")
    // <editor-fold defaultstate="collapsed" desc="Generated Code">
    private void initComponents() {

        jLabel1 = new javax.swing.JLabel();
        jTabbedPane1 = new javax.swing.JTabbedPane();
        jPanel1 = new javax.swing.JPanel();
        jLabel2 = new javax.swing.JLabel();
        campoIPServidor = new javax.swing.JTextField();
        jLabel3 = new javax.swing.JLabel();
        campoPuertoServidor = new javax.swing.JTextField();
        btnIniciar = new javax.swing.JButton();
        jLabel4 = new javax.swing.JLabel();
        txtEstado = new javax.swing.JLabel();
        jPanel3 = new javax.swing.JPanel();
        jLabel5 = new javax.swing.JLabel();
        campoPeso = new javax.swing.JTextField();
        jLabel6 = new javax.swing.JLabel();
    }
}
```

Guía básica de Desarrollo de sistemas distribuidos usando Socket TCP y Java.

Autor: John Carlos Arrieta Arrieta

```
campoAltura = new javax.swing.JTextField();
btnIniciar1 = new javax.swing.JButton();
jLabel7 = new javax.swing.JLabel();
txtResultado = new javax.swing.JLabel();
txtMensaje = new javax.swing.JLabel();

setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);

jLabel1.setFont(new java.awt.Font("Tahoma", 0, 24)); // NOI18N
jLabel1.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
jLabel1.setText("CLIENTE IMC");

jLabel2.setText("DIRECCION IP: ");

campoIPServidor.setText("localhost");

jLabel3.setText("PUERTO DE RED:");

campoPuertoServidor.setText("9007");

btnIniciar.setFont(new java.awt.Font("Tahoma", 1, 14)); // NOI18N
btnIniciar.setForeground(new java.awt.Color(0, 153, 51));
btnIniciar.setText("Conectar");
btnIniciar.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnIniciarActionPerformed(evt);
    }
});

jLabel4.setText("ESTADO: ");

txtEstado.setForeground(new java.awt.Color(255, 0, 51));
txtEstado.setText("Desconectado");

javax.swing.GroupLayout jPanel1Layout = new
javax.swing.GroupLayout(jPanel1);
jPanel1.setLayout(jPanel1Layout);
jPanel1Layout.setHorizontalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(jPanel1Layout.createSequentialGroup()
        .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
            .addGroup(jPanel1Layout.createSequentialGroup()
                .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
                    .addGroup(jPanel1Layout.createSequentialGroup()
                        .addComponent(jLabel2,
```

Guía básica de Desarrollo de sistemas distribuidos usando Socket TCP y Java.

Autor: John Carlos Arrieta Arrieta

```
javax.swing.GroupLayout.PREFERRED_SIZE, 131,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
    .addComponent(campoIPServidor,
javax.swing.GroupLayout.PREFERRED_SIZE, 152,
javax.swing.GroupLayout.PREFERRED_SIZE))
    .addGroup(jPanel1Layout.createSequentialGroup())
    .addComponent(jLabel3,
javax.swing.GroupLayout.PREFERRED_SIZE, 131,
javax.swing.GroupLayout.PREFERRED_SIZE)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
    .addComponent(campoPuertoServidor))
    .addGroup(jPanel1Layout.createSequentialGroup())
    .addComponent(jLabel4)

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
    .addComponent(txtEstado,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
    .addComponent(btnIniciar,
javax.swing.GroupLayout.Alignment.TRAILING,
javax.swing.GroupLayout.PREFERRED_SIZE, 145,
javax.swing.GroupLayout.PREFERRED_SIZE))
    .addContainerGap(137, Short.MAX_VALUE))
);
jPanel1Layout.setVerticalGroup(

jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(jPanel1Layout.createSequentialGroup())
    .addGap(21, 21, 21)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
    .addComponent(jLabel2)
    .addComponent(campoIPServidor,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
    .addGap(18, 18, 18)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
    .addComponent(jLabel3)
    .addComponent(campoPuertoServidor,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
```

Guía básica de Desarrollo de sistemas distribuidos usando Socket TCP y Java.

Autor: John Carlos Arrieta Arrieta

```
        .addGap(18, 18, 18)

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BA
SELINE)
        .addComponent(jLabel14)
        .addComponent(txtEstado))

.addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 19,
Short.MAX_VALUE)
        .addComponent(btnIniciar)
        .addGap(20, 20, 20))
);

jTabbedPane1.addTab("CONEXION", jPanel1);

jLabel5.setText("PESO:");

jLabel6.setText("ALTURA:");

btnIniciar1.setFont(new java.awt.Font("Tahoma", 1, 14)); // NOI18N
btnIniciar1.setForeground(new java.awt.Color(0, 153, 51));
btnIniciar1.setText("CALCULAR");
btnIniciar1.addActionListener(new java.awt.event.ActionListener() {
    public void actionPerformed(java.awt.event.ActionEvent evt) {
        btnIniciar1ActionPerformed(evt);
    }
});

jLabel7.setText("IMC: ");

txtResultado.setFont(new java.awt.Font("Tahoma", 1, 12)); // NOI18N
txtResultado.setForeground(new java.awt.Color(255, 0, 51));
txtResultado.setText("0.0");

txtMensaje.setBorder(javax.swing.BorderFactory.createTitledBorder(""));

javax.swing.GroupLayout jPanel3Layout = new
javax.swing.GroupLayout(jPanel3);
jPanel3.setLayout(jPanel3Layout);
jPanel3Layout.setHorizontalGroup(

jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(jPanel3Layout.createSequentialGroup()
        .addContainerGap()

.addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LE
ADING)
```

Guía básica de Desarrollo de sistemas distribuidos usando Socket TCP y Java.

Autor: John Carlos Arrieta Arrieta

```
                .addGroup(jPanel3Layout.createSequentialGroup()  
                    .addComponent(jLabel17)  
                )  
  
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)  
            .addComponent(txtResultado,  
javax.swing.GroupLayout.PREFERRED_SIZE, 66,  
javax.swing.GroupLayout.PREFERRED_SIZE)  
  
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)  
            .addComponent(txtMensaje,  
javax.swing.GroupLayout.PREFERRED_SIZE, 285,  
javax.swing.GroupLayout.PREFERRED_SIZE)  
            .addGap(0, 0, Short.MAX_VALUE))  
            .addGroup(jPanel3Layout.createSequentialGroup()  
  
        .addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
            .addGroup(jPanel3Layout.createSequentialGroup()  
                .addComponent(jLabel6,  
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,  
Short.MAX_VALUE)  
  
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)  
            .addComponent(campoAltura,  
javax.swing.GroupLayout.PREFERRED_SIZE, 152,  
javax.swing.GroupLayout.PREFERRED_SIZE))  
            .addGroup(jPanel3Layout.createSequentialGroup()  
                .addComponent(jLabel15,  
javax.swing.GroupLayout.PREFERRED_SIZE, 66,  
javax.swing.GroupLayout.PREFERRED_SIZE)  
  
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)  
            .addComponent(campoPeso,  
javax.swing.GroupLayout.PREFERRED_SIZE, 152,  
javax.swing.GroupLayout.PREFERRED_SIZE)))  
            .addGap(29, 29, 29)  
            .addComponent(btnIniciar1,  
javax.swing.GroupLayout.PREFERRED_SIZE, 145,  
javax.swing.GroupLayout.PREFERRED_SIZE))  
            .addGap(28, 28, 28))  
        );  
        jPanel3Layout.setVerticalGroup(  
  
jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)  
    .addGroup(jPanel3Layout.createSequentialGroup()  
        .addGap(21, 21, 21)
```

Guía básica de Desarrollo de sistemas distribuidos usando Socket TCP y Java.

Autor: John Carlos Arrieta Arrieta

```
.addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
    .addGroup(jPanel3Layout.createSequentialGroup()
        .addComponent(jLabel15)
        .addPreferredGap(LayoutStyle.ComponentPlacement.RELATED)
        .addComponent(campoPeso,
            javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
            javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGap(28, 28, 28)
    .addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
        .addComponent(jLabel16)
        .addComponent(campoAltura,
            javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
            javax.swing.GroupLayout.PREFERRED_SIZE))
        .addGap(21, 21, 21)
        .addGroup(jPanel3Layout.createSequentialGroup()
            .addComponent(btnIniciar1,
                javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
                Short.MAX_VALUE)
            .addGap(18, 18, 18)))
    .addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
        .addComponent(jLabel17)
        .addComponent(txtResultado,
            javax.swing.GroupLayout.PREFERRED_SIZE, 22,
            javax.swing.GroupLayout.PREFERRED_SIZE)
        .addComponent(txtMensaje,
            javax.swing.GroupLayout.PREFERRED_SIZE, 22,
            javax.swing.GroupLayout.PREFERRED_SIZE))
        .addContainerGap(43, Short.MAX_VALUE))
);

jTabbedPane1.addTab("CALCULAR IMC", jPanel3);

javax.swing.GroupLayout layout = new
javax.swing.GroupLayout(getContentPane());
getContentPane().setLayout(layout);
layout.setHorizontalGroup(
    layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
        .addGroup(layout.createSequentialGroup()
            .addContainerGap()
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jLabel15)
                .addComponent(campoPeso,
                    javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
                    javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(28, 28, 28)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(jLabel16)
                .addComponent(campoAltura,
                    javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
                    javax.swing.GroupLayout.PREFERRED_SIZE))
            .addGap(21, 21, 21)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(btnIniciar1,
                    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
                    Short.MAX_VALUE)
                .addGap(18, 18, 18))
            .addGap(43, 43, 43)
            .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
                .addComponent(txtResultado,
                    javax.swing.GroupLayout.PREFERRED_SIZE, 22,
                    javax.swing.GroupLayout.PREFERRED_SIZE)
                .addComponent(txtMensaje,
                    javax.swing.GroupLayout.PREFERRED_SIZE, 22,
                    javax.swing.GroupLayout.PREFERRED_SIZE))
            .addContainerGap())
    );
}
```

Guía básica de Desarrollo de sistemas distribuidos usando Socket TCP y Java.

Autor: John Carlos Arrieta Arrieta

```
.addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addComponent(jLabel1,
javax.swing.GroupLayout.PREFERRED_SIZE, 437,
javax.swing.GroupLayout.PREFERRED_SIZE)
    .addComponent(jTabbedPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.PREFERRED_SIZE))
    .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE))
);
layout.setVerticalGroup(

layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
    .addGroup(layout.createSequentialGroup()
        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
            .add(layout.createSequentialGroup()
                .addGap(27, 27, 27)
                .addComponent(jLabel1)
                .addGap(32, 32, 32)
                .addComponent(jTabbedPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 203,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(37, Short.MAX_VALUE))
            .add(layout.createSequentialGroup()
                .addGap(27, 27, 27)
                .addComponent(jLabel1)
                .addGap(32, 32, 32)
                .addComponent(jTabbedPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 203,
javax.swing.GroupLayout.PREFERRED_SIZE)
                .addContainerGap(37, Short.MAX_VALUE))
        )
    )
);

pack();
} // </editor-fold>

private void btnIniciarActionPerformed(java.awt.event.ActionEvent evt) {
    int puerto = Integer.parseInt(campoPuertoServidor.getText());
    String ip = campoIPServidor.getText();
    try {
        if (btnIniciar.getText().equalsIgnoreCase("Conectar")) {
            servidor = new Socket(ip, puerto);
            out = new DataOutputStream(servidor.getOutputStream());
            in = new DataInputStream(servidor.getInputStream());
            btnIniciar.setText("Desconectar");
            btnIniciar.setForeground(Color.RED);
            txtEstado.setText("Conectado");
            txtEstado.setForeground(Color.GREEN);
        } else if (btnIniciar.getText().equalsIgnoreCase("Desconectar")) {
            if (servidor.isConnected()) {
                servidor.close();
            }
            btnIniciar.setText("Conectar");
            txtEstado.setText("Desconectado");
            btnIniciar.setForeground(Color.GREEN);
            txtEstado.setForeground(Color.RED);
        }
    } catch (IOException ex) {
        JOptionPane.showMessageDialog(this, ex.getMessage(), "Error",
JOptionPane.ERROR_MESSAGE);
    }
}
```

Guía básica de Desarrollo de sistemas distribuidos usando Socket TCP y Java.

Autor: John Carlos Arrieta Arrieta

```
    }
    } catch (IOException ex) {
        System.out.println("ERROR AL CONECTAR");
        ex.printStackTrace();
    }
}

private void btnIniciarActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    if (!servidor.isConnected()) {
        JOptionPane.showConfirmDialog(this, "Cliente OffLine, Conecte con el
Servidor");
        return;
    }
    float peso = Float.parseFloat(campoPeso.getText());
    float altura = Float.parseFloat(campoAltura.getText());
    Thread hilo = new Thread() {
        @Override
        public void run() {
            try {
                System.out.println("Peso: " + peso);
                System.out.println("Altura: " + altura);
                out.writeFloat(peso);
                out.writeFloat(altura);
                out.flush();
                System.out.println("Enviados los datos\nEsperando
respuesta");

                float imc = in.readFloat();
                String msj = in.readUTF();
                System.out.println("IMC: " + imc + "\nMensaje: " + msj);
                txtResultado.setText(imc + "");
                txtMensaje.setText(msj);
            } catch (IOException ex) {
                JOptionPane.showMessageDialog(VentanaPrincipal.this, "ERROR
con el cliente " + ex.getMessage());
                System.out.println("ERROR con el cliente " +
ex.getMessage());
                ex.printStackTrace();
            }
        }
    };
    hilo.start();
}

public JLabel getTxtEstado() {
    return txtEstado;
}
```


Guía básica de Desarrollo de sistemas distribuidos usando Socket TCP y Java.

Autor: John Carlos Arrieta Arrieta

```
public JButton getBtnIniciar() {
    return btnIniciar;
}

/**
 * @param args the command line arguments
 */
public static void main(String args[]) {
    /* Set the Nimbus Look and feel */
    //<editor-fold defaultstate="collapsed" desc=" Look and feel setting
code (optional) ">
    /* If Nimbus (introduced in Java SE 6) is not available, stay with the
default look and feel.
    * For details see
http://download.oracle.com/javase/tutorial/uiswing/lookandfeel/plaf.html
    */
    try {
        for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {
                javax.swing.UIManager.setLookAndFeel(info.getClassName());
                break;
            }
        }
    } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(VentanaPrincipal.class.getName()).log(java.ut
il.logging.Level.SEVERE, null, ex);
    } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(VentanaPrincipal.class.getName()).log(java.ut
il.logging.Level.SEVERE, null, ex);
    } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(VentanaPrincipal.class.getName()).log(java.ut
il.logging.Level.SEVERE, null, ex);
    } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(VentanaPrincipal.class.getName()).log(java.ut
il.logging.Level.SEVERE, null, ex);
    }
    //</editor-fold>
    //</editor-fold>

    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable() {
```

Guía básica de Desarrollo de sistemas distribuidos usando Socket TCP y Java.

Autor: John Carlos Arrieta Arrieta

```
        public void run() {
            new VentanaPrincipal().setVisible(true);
        }
    });
}

// Variables declaration - do not modify
private javax.swing.JButton btnIniciar;
private javax.swing.JButton btnIniciar1;
private javax.swing.JTextField campoAltura;
private javax.swing.JTextField campoIPServidor;
private javax.swing.JTextField campoPeso;
private javax.swing.JTextField campoPuertoServidor;
private javax.swing.JLabel jLabel1;
private javax.swing.JLabel jLabel2;
private javax.swing.JLabel jLabel3;
private javax.swing.JLabel jLabel4;
private javax.swing.JLabel jLabel5;
private javax.swing.JLabel jLabel6;
private javax.swing.JLabel jLabel7;
private javax.swing.JPanel jPanel1;
private javax.swing.JPanel jPanel3;
private javax.swing.JTabbedPane jTabbedPane1;
private javax.swing.JLabel txtEstado;
private javax.swing.JLabel txtMensaje;
private javax.swing.JLabel txtResultado;
// End of variables declaration
}
```

Este es el archivo pirincipal que ejecuta la ventana principal del cliente

ARCHIVO PRINCPAL.JAVA

```
package johnarrieta.imc.cliente;

import johnarrieta.imc.cliente.vistas.VentanaPrincipal;

/**
 *
 * @author JOHN CARLOS ARRIETA ARRIETA
 */
public class Principal {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        VentanaPrincipal v = new VentanaPrincipal();
    }
}
```

Guía básica de Desarrollo de sistemas distribuidos usando Socket TCP y Java.

Autor: John Carlos Arrieta Arrieta

```
        v.setLocationRelativeTo(null);  
        v.setVisible(true);  
    }  
  
}
```

Guía básica de Desarrollo de sistemas distribuidos usando Socket TCP y Java.

Autor: John Carlos Arrieta Arrieta

Aqui realizamos las pruebas iniciando el servidor y conectándonos con el cliente atravez de una dirección ip y un puerto tcp

LAS PRUEBAS

The image displays three screenshots of Java Swing windows used for testing a distributed system. The top row shows two windows side-by-side: 'CLIENTE IMC' and 'SERVIDOR IMC'. The 'CLIENTE IMC' window has tabs for 'CONEXION' and 'CALCULAR IMC'. Under 'CONEXION', it has input fields for 'DIRECCION IP:' (containing 'localhost') and 'PUERTO DE RED:' (containing '9007'). The status is 'ESTADO: Desconectado' in red, and there is a green 'Conectar' button. The 'SERVIDOR IMC' window has tabs for 'CONEXION' and 'LOG DE CONEXIONES'. Under 'CONEXION', it has input fields for 'DIRECCION IP:' (containing '192.168.1.16') and 'PUERTO DE RED:' (containing '9007'). The status is 'ESTADO: DETENIDO' in red, and there is a green 'INICIAR' button. The bottom screenshot is a larger view of the 'SERVIDOR IMC' window, showing the 'CONEXION' tab selected. The status is now 'ESTADO: ONLINE' in green, and the button is red and labeled 'DETENER'.

CLIENTE IMC

CONEXION CALCULAR IMC

DIRECCION IP: localhost

PUERTO DE RED: 9007

ESTADO: Desconectado

Conectar

SERVIDOR IMC

CONEXION LOG DE CONEXIONES

DIRECCION IP: 192.168.1.16

PUERTO DE RED: 9007

ESTADO: DETENIDO

INICIAR

SERVIDOR IMC

CONEXION LOG DE CONEXIONES

DIRECCION IP: 192.168.1.16

PUERTO DE RED: 9007

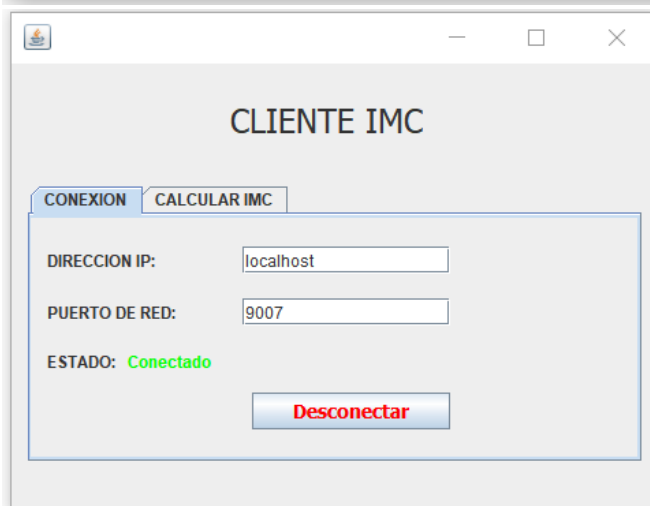
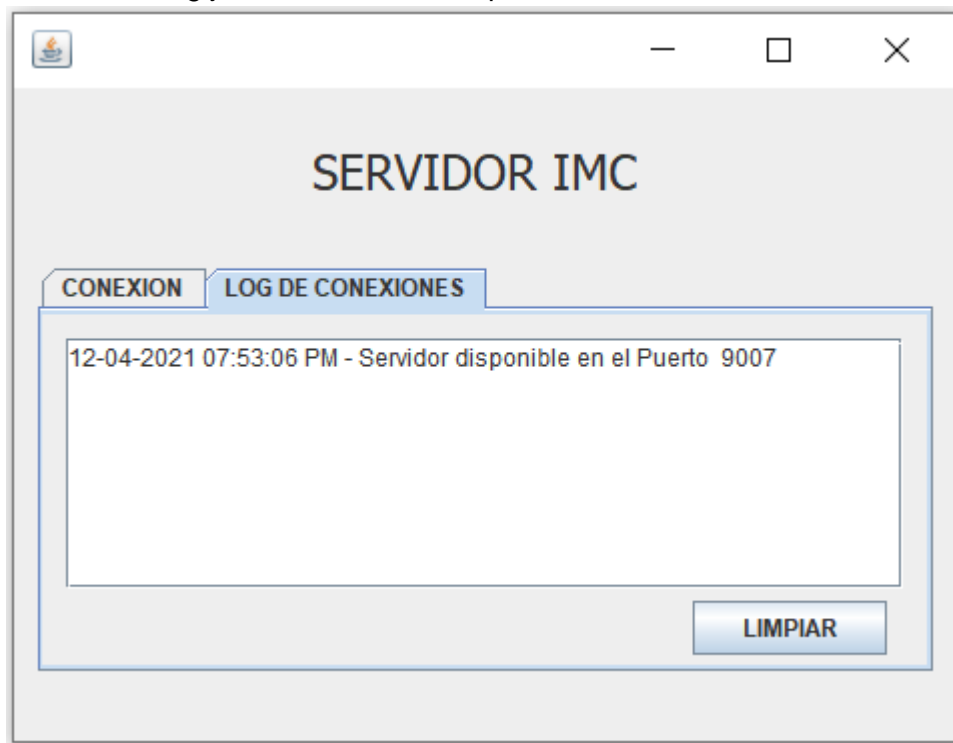
ESTADO: ONLINE

DETENER

Guía básica de Desarrollo de sistemas distribuidos usando Socket TCP y Java.

Autor: John Carlos Arrieta Arrieta

Aquí se muestra el log junto con un cálculo que se le hizo al servidor



Guía básica de Desarrollo de sistemas distribuidos usando Socket TCP y Java.

Autor: John Carlos Arrieta Arrieta

```
12-04-2021 07:53:06 PM - Servidor disponible en el Puerto 9007
12-04-2021 07:54:34 PM - Cliente 127.0.0.1 conectado
```

```
127.0.0.1 -> 12-04-2021 07:54:34 PM - Esperando el PESO:
127.0.0.1 -> 12-04-2021 07:56:08 PM - PESO: 81.0
127.0.0.1 -> 12-04-2021 07:56:08 PM - Esperando La Altura:
127.0.0.1 -> 12-04-2021 07:56:08 PM - ALTURA: 1.7
127.0.0.1 -> 12-04-2021 07:56:08 PM - IMC: 28.027681
127.0.0.1 -> 12-04-2021 07:56:08 PM - IMC: 28.027681
127.0.0.1 -> 12-04-2021 07:56:08 PM - MENSAJE: Debes bajar un poco de peso
127.0.0.1 -> 12-04-2021 07:56:08 PM - MENSAJE: Debes bajar un poco de peso
127.0.0.1 -> 12-04-2021 07:56:08 PM - IMC: 28.027681
127.0.0.1 -> 12-04-2021 07:56:08 PM - MENSAJE: Debes bajar un poco de peso
127.0.0.1 -> 12-04-2021 07:56:08 PM - Esperando el PESO:
```

The image shows two side-by-side Java Swing windows. The left window, titled 'CLIENTE IMC', has two tabs: 'CONEXION' and 'CALCULAR IMC'. The 'CALCULAR IMC' tab is active, showing input fields for 'PESO:' (81) and 'ALTURA:' (1.7), a 'CALCULAR' button, and a result area for 'IMC:' displaying '28.0276...' and a message 'Debes bajar un poco de peso'. The right window, titled 'SERVIDOR IMC', has two tabs: 'CONEXION' and 'LOG DE CONEXIONES'. The 'LOG DE CONEXIONES' tab is active, displaying a scrollable log of the client's interactions, including the same messages seen in the terminal output above. A 'LIMPIAR' button is located at the bottom right of the log area.