Autor: John Carlos Arrieta Arrieta

FOTO GUIA SOBRE DESARROLLO DE APLICACIONES DISTRIBUIDAS CLIENTE SERVIDOR SOBRE TCP/IP USANDO TECNOLOGÍA RMI (REMOTE METHOD INVOCATION) DE JAVA Y UNA LIBRERÍA DE TERCEROS LLAMADA LIPERMI QUE FACILITA EL PROCESO DE IMPLEMENTACIÓN DE RMI

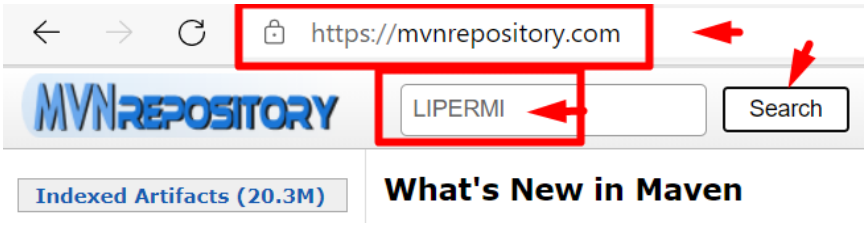
Como actividad académica los estudiantes de forma individual deben:

1. Realizar y documentar cada paso de esta Guía, entregando los fuentes de los 3 proyectos
2. Realizar el mismo ejemplo de la guía usando RMI Estándar de Java, sin usar la librería LipeRM, documentando cada paso del proceso y explicando las diferencias, similitudes con el uso de la LipeRmi.
3. Investigar si existen otras librerías (en Java, o entros lenguajes) que permitan desarrollar Sistemas Distribuidos usando RMI/RPC y entregar un ejemplo muy simple.

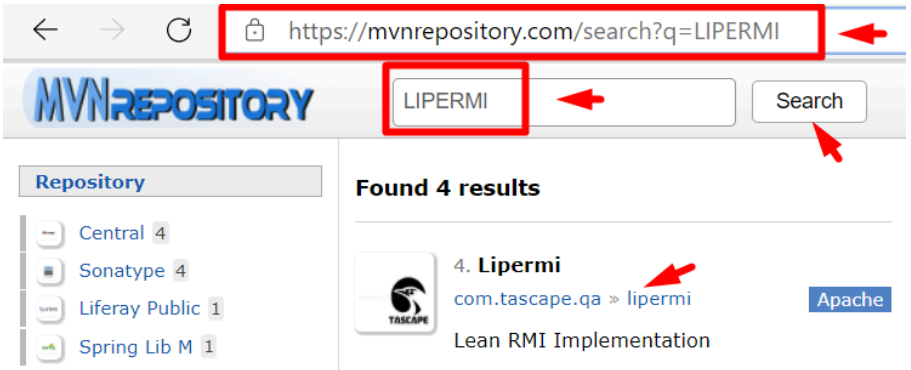
DESCARGAR, INSTALAR Y CONFIGURAR LAS HERRAMIENTAS NECESARIAS

a) Descargar, Instalar y configurar correctamente el JDK
   i) https://youtu.be/BoYRcKZbDb0
b) Descargar un IDE para Java, por ejemplo Netbeans IDE
   i) https://youtu.be/oT1cUI984zU
c) Descargar la librería LipeRMI https://mvnrepository.com/

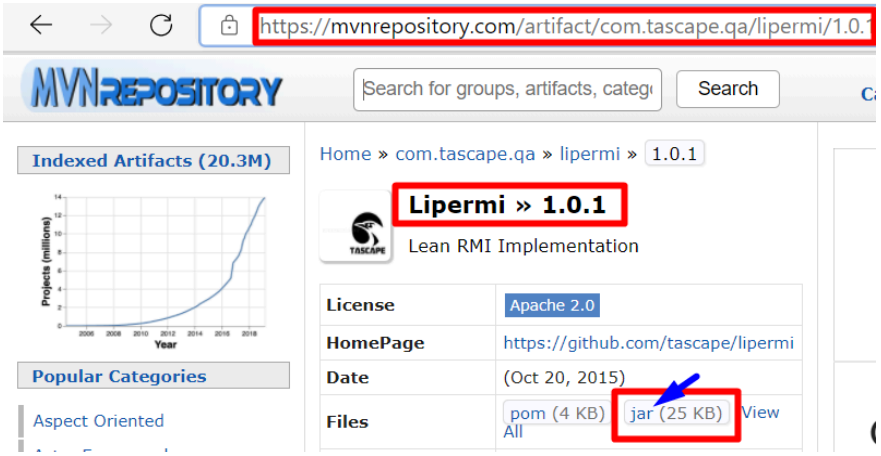
PASOS:



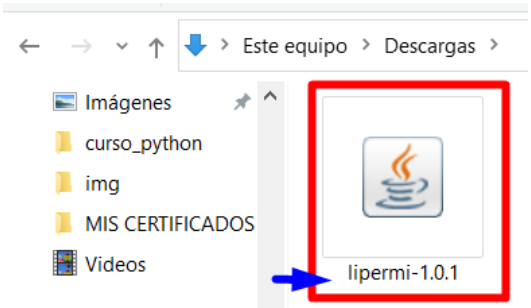
buscar la página web mvnrepository, en su buscador poner LIPERMI



seleccionar la opción que solo tenga en su nombre LIPERMI; exactamente en el link debajo del nombre.

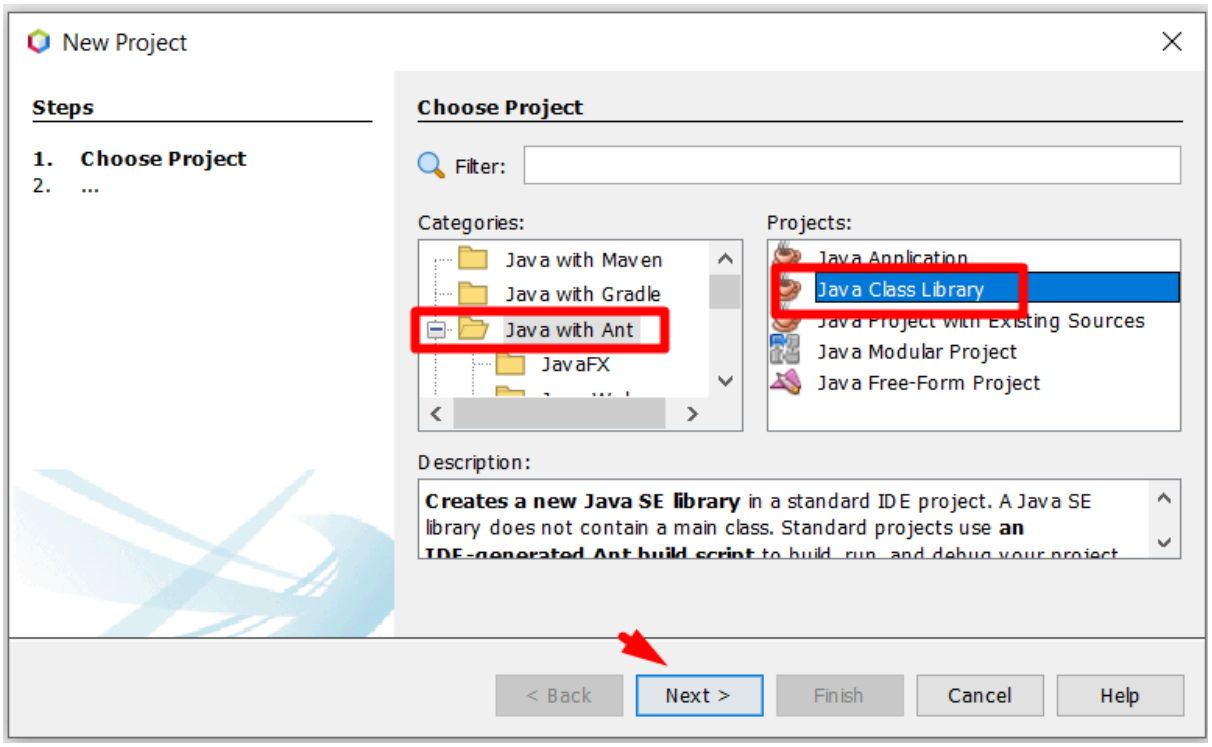


En esta nueva ventana hay que rectificar que la versión de lipermi sea la 1.0.1, además hay que descargar la versión del archivo que tenga la extensión .jar.

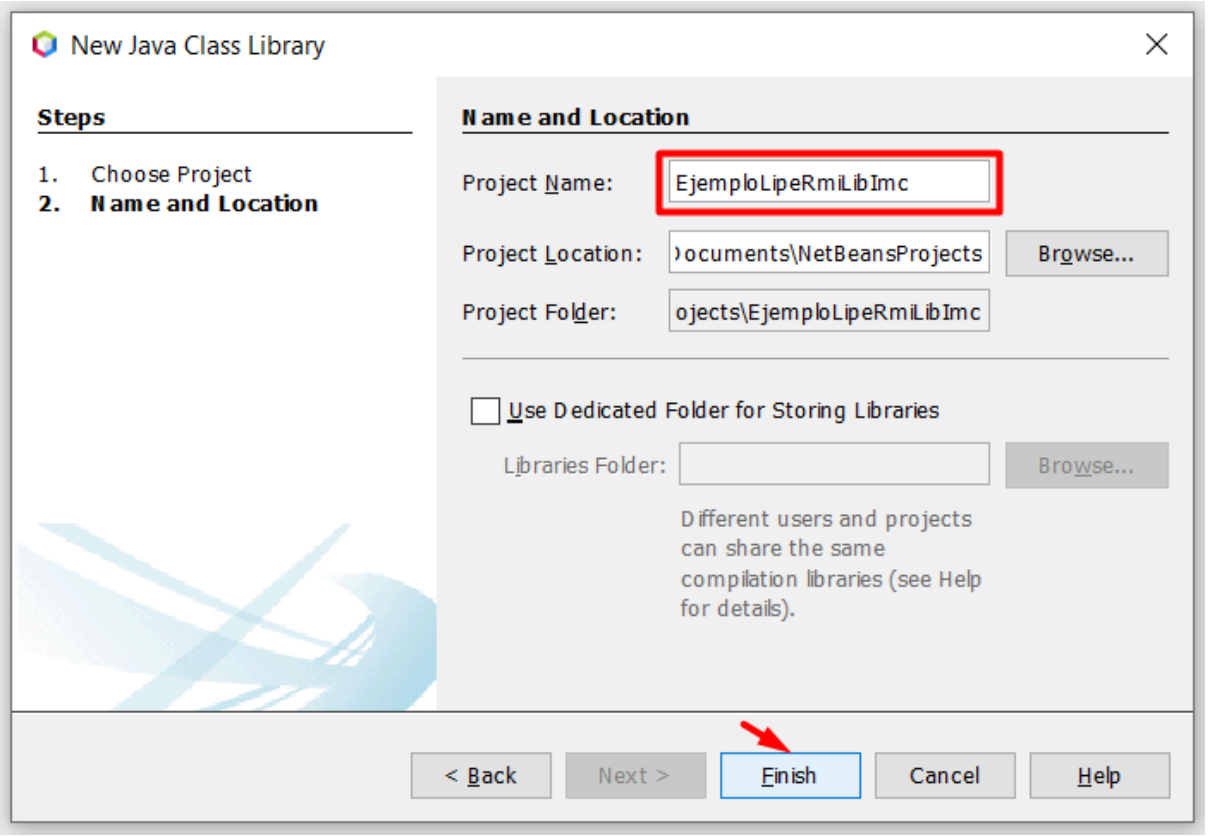


Verificar que la versión que instalaste sea la correcta.

CREAR UN PROYECTO TIPO LIBRERÍA

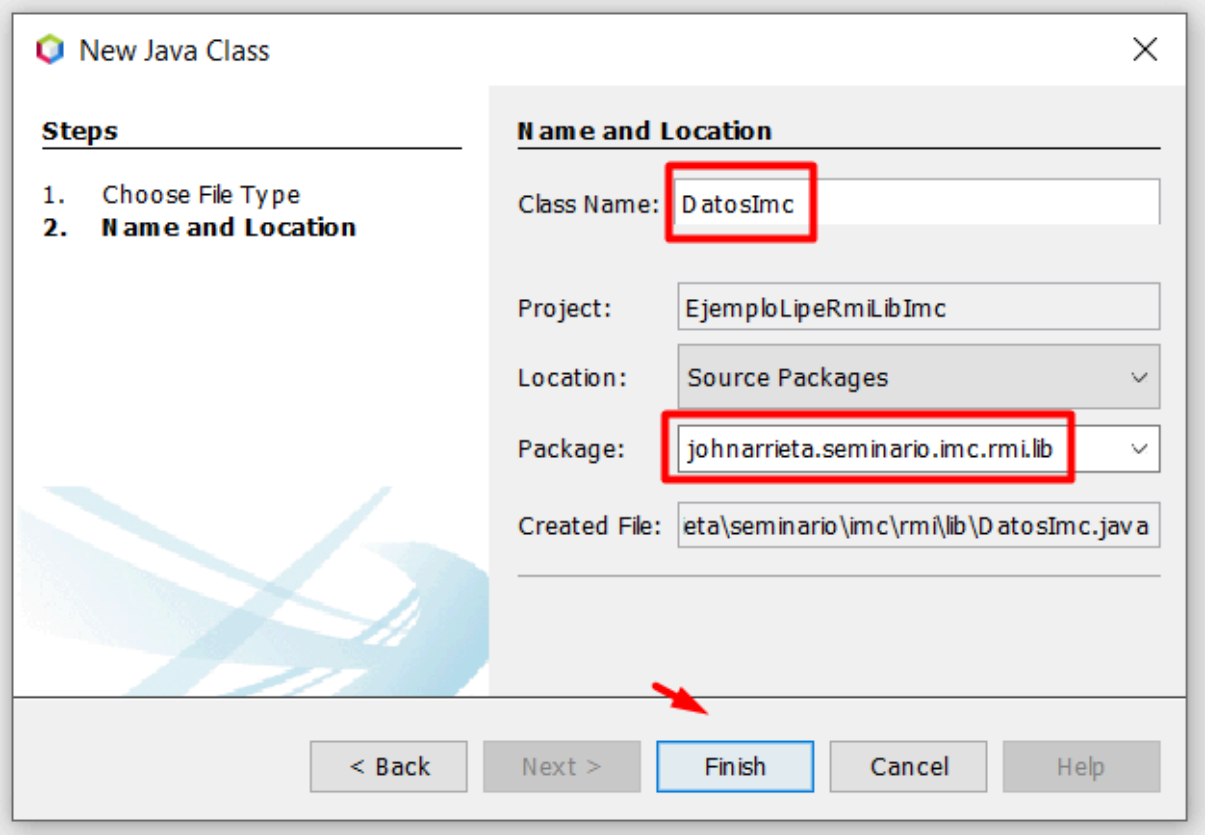


1. Creamos un nuevo proyecto, damos click en el botón new project,
2. En esta nueva ventana buscamos la carpeta java with ant, y dentro elegimos la opción java class library.
3. damos click en next.



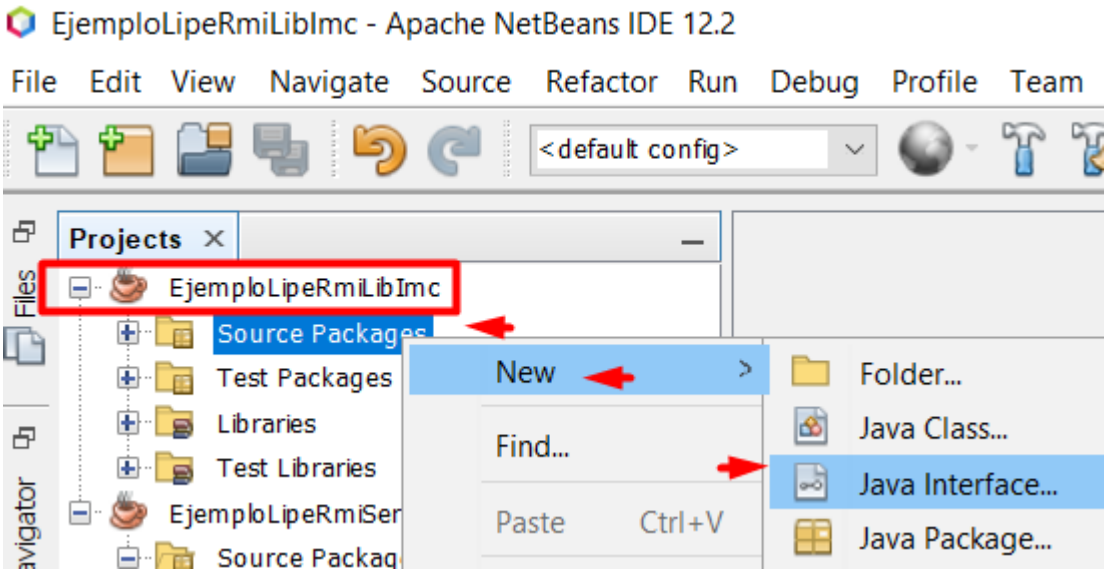
1. En esta nueva ventana “NAME and Location”, colocamos el nombre del proyecto.
2. Ponemos la ubicación que queramos.
3. Damos clic en el botón finish.

2) Crear una clase para transportar los datos del resultado del Imc

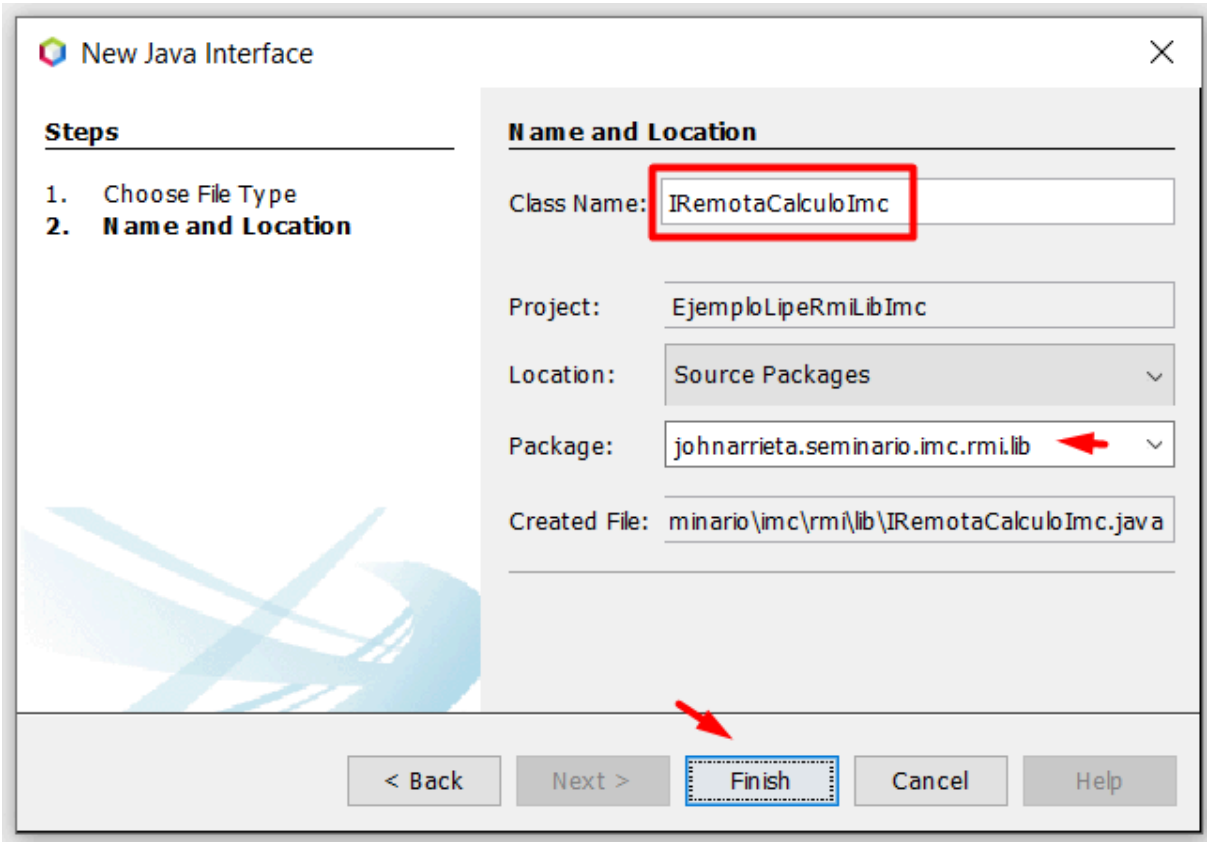


1. Seleccionamos el nombre del proyecto y damos clic derecho, saldrá una opción que dice new, se da clic hay y después se da clic en java class.
2. en esta nueva ventana le damos nombre a la clase.
3. le creamos un paquete para clase con su respectivo nombre.
4. Finalizamos todo dando clic en el botón finish.

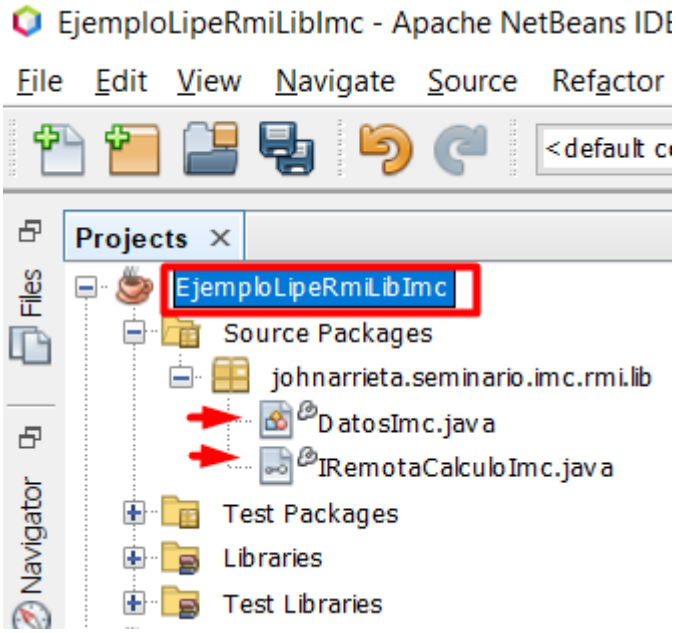
3) Crear una interfaz para invocar los métodos de forma remota



1. Clic derecho en el paquete “source packages”.
2. Clic en new; seleccionamos java interfaz y damos clic en hay.



1. En esta nueva ventana le damos nombre a la interfaz en Class Name.
2. le asignamos el mismo paquete que la clase anterior.
3. Finalizamos todo dando clic en el botón finish.



verificamos que todo fue creado con exito.

CÓDIGO DE LA CLASE **DATOSIMC**

```java
package johnarrieta.seminario.imc.rmi.lib;

import java.io.Serializable;

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

    private float peso;
    private float altura;
    private float resultado;
    private String interpretación;

    public DatosImc() {
    }

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

    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;
    }

    public float getResultado() {
        return resultado;
    }

    public void setResultado(float resultado) {
        this.resultado = resultado;
    }

    public String getInterpretacion() {
        return interpretacion;
    }

    public void setInterpretacion(String interpretacion) {
        this.interpretacion = interpretacion;
    }

}
```

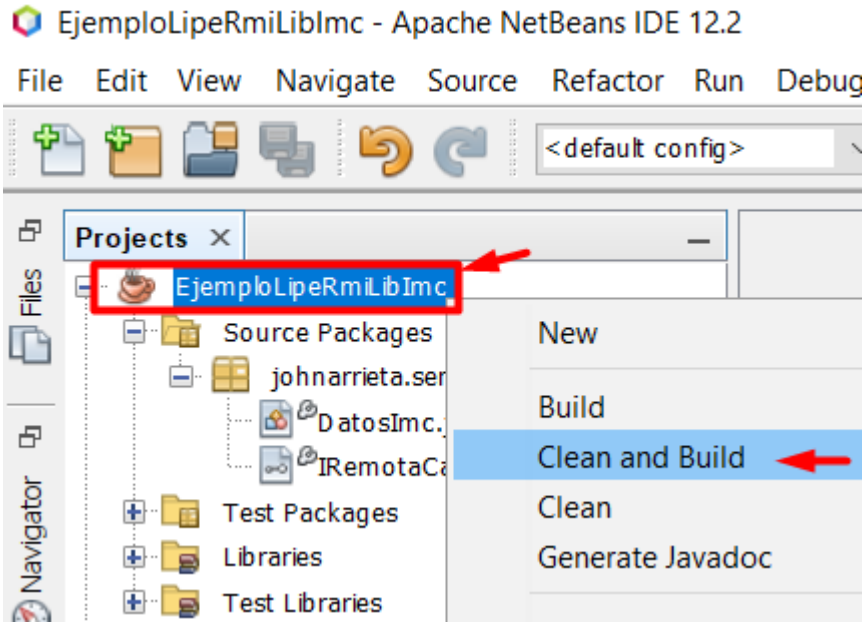
En primera instancia se importa la clase serializable y se agrega al lado del nombre de la clase como implements serializable.
La clase tiene 4 atributos: peso, altura, resultado e interpretación, además, la clase tiene 2 constructores; uno vacío y, el otro solo acepta peso y altura.
La clase tiene varios métodos accesores y mutadores de los atributos de la clase.

CÓDIGO DE LA INTERFAZ IREMOTACALCULOIMC

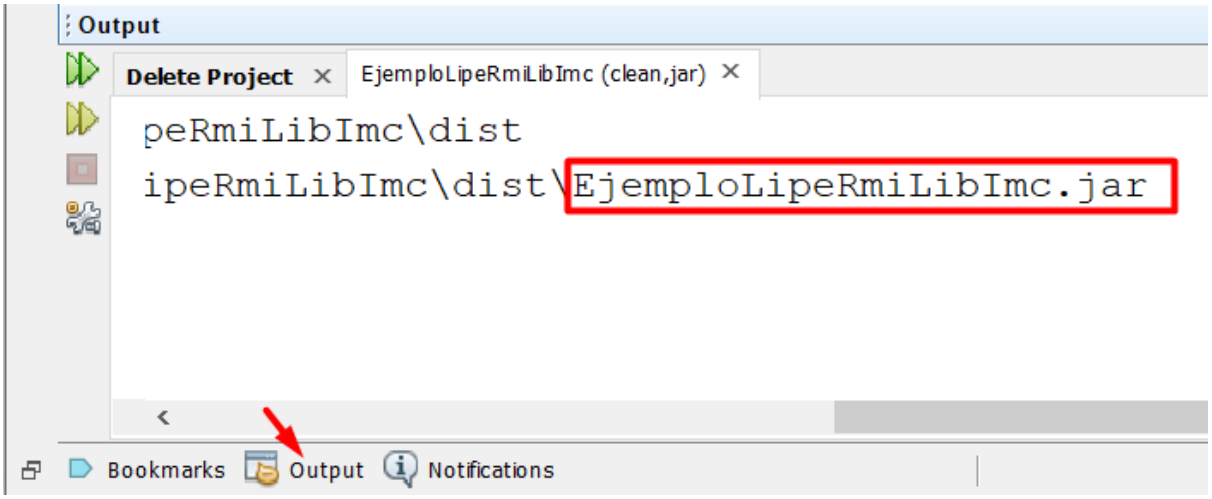
```java
package johnarrieta.seminario.imc.rmi.lib;

/**
 *
 * @author  JOHN CARLOS ARRIETA ARRIETA
 */
public interface IRemotaCalculoImc {

    public DatosImc calcularImc(DatosImc datos);
}
```

En la interfaz se crea un método clacularImc, además se tiene que ingresar un parámetro, que es tipo de la clase antes creada(DatosImc).

5. Construir el archivo distribuible (librerías, dependencia o componente JAR)



se da clic derecho al nombre del proyecto, y damos clic otra vez en clean and build



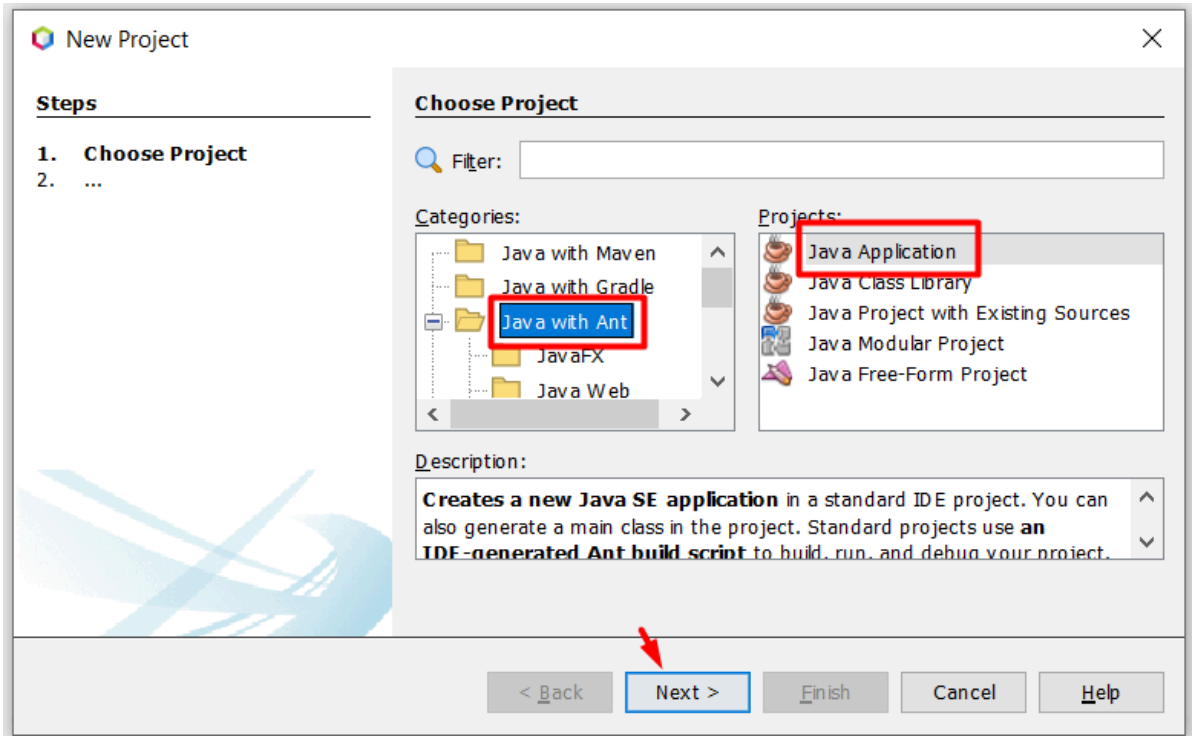
Vemos que la aplicación netbeans crear un .jar del proyecto.

CREAR EL PROYECTO PARA EL SERVIDOR

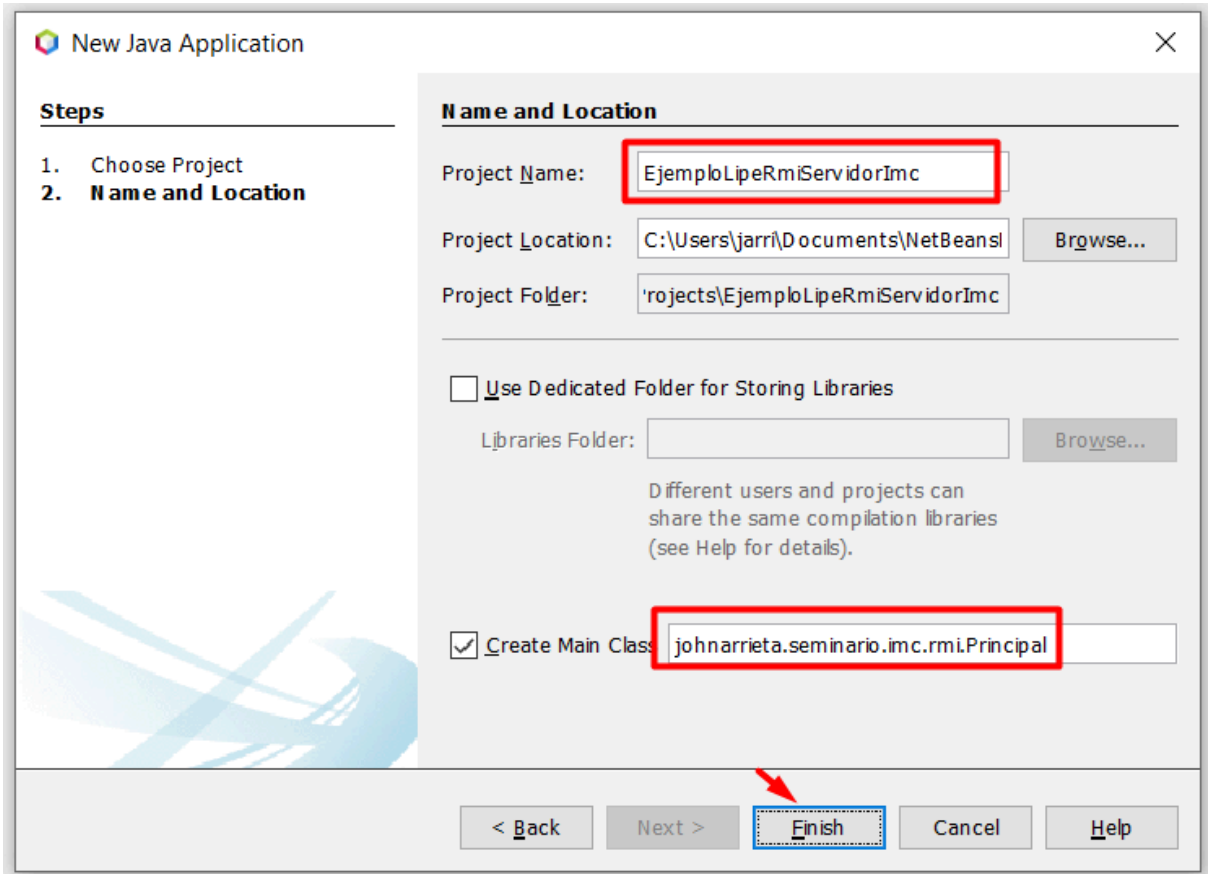
1.Crear un nuevo proyecto de tipo Java Application

Pasos:

1. Crear un nuevo proyecto en NB
2. Categoría Proyecto con Ant
3. Tipo Java Application
4. Nombre: EjemploLipeRmiServidorImc
5. Ubicación: Cualquier carpeta
6. Clase Principal: Tu_nombre.seminario.imc.rmi.Principal

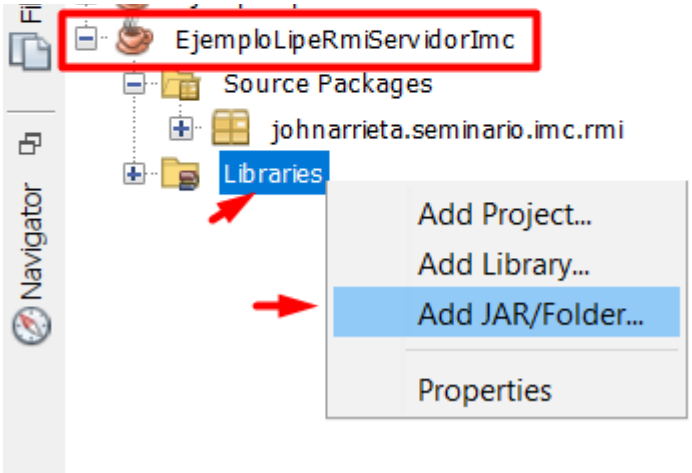


1. Creamos un nuevo proyecto, damos click en el botón new project,
2. En esta nueva ventana buscamos la carpeta java with ant, y dentro elegimos la opción java Aplication.
3. damos click en next.

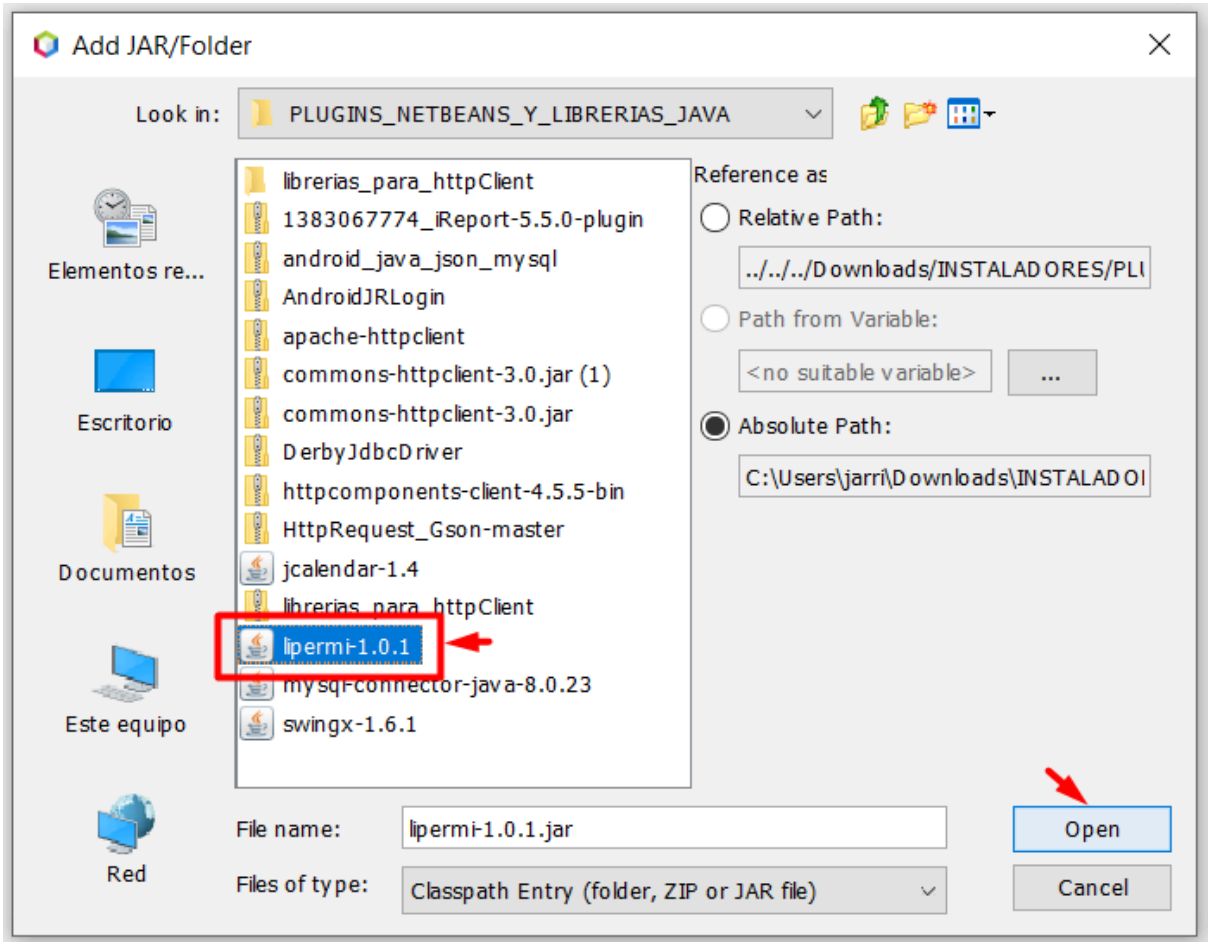


1. En esta nueva ventana “NAME and Location”, colocamos el nombre del proyecto.
2. Ponemos la ubicación que queramos, en el botón Browser.
3. agregamos enseguida la clase principal de manera automática.
4. Damos clic en el botón finish.

2. Agregar la librería (componente o dependencia) LipeRMI

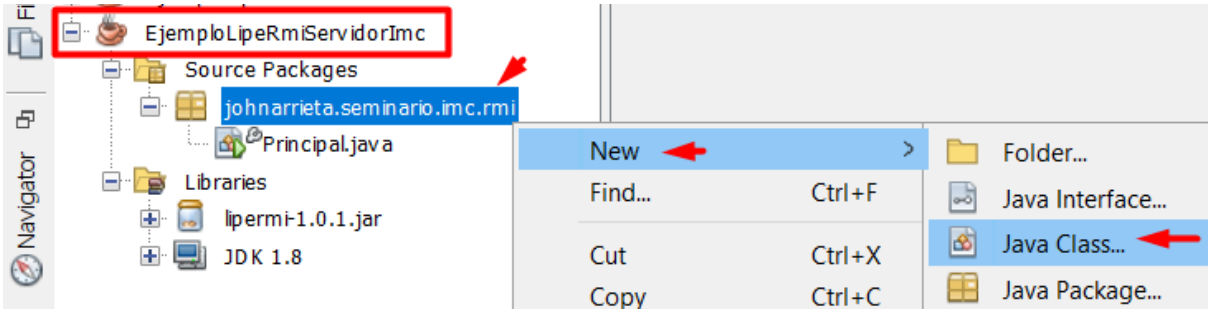


Ahora agregamos la librería LipeRMI, damos clic derecho en la carpeta librerías; después click en la opción add JAR/Folder.

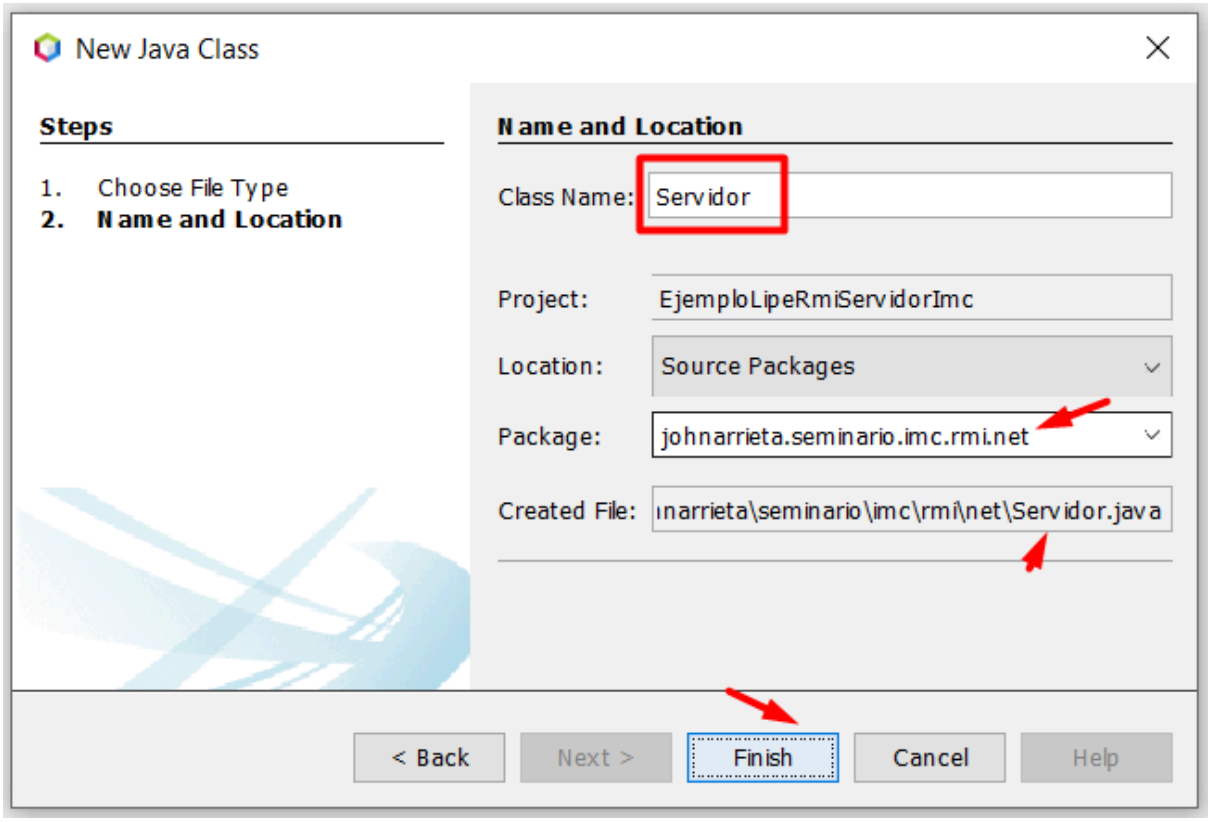


en la ventana add JAR/Folder; agregamos la librería lipermi, dando clic en el archivo y después click en el botón open.

3. Crear la clase Servidor

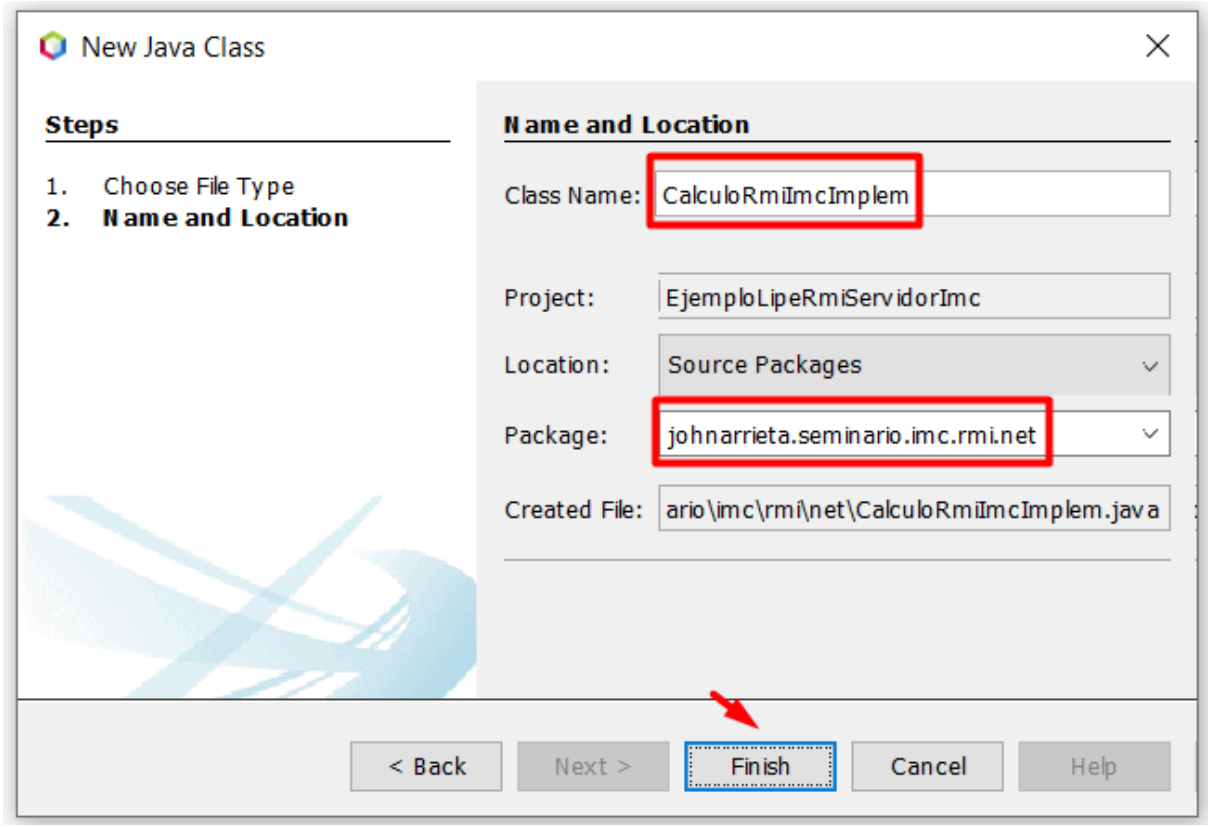


Ahora vamos a crear la clase del servidor; click derecho en la única carpeta del proyecto, seleccionamos new y damos clic en java class

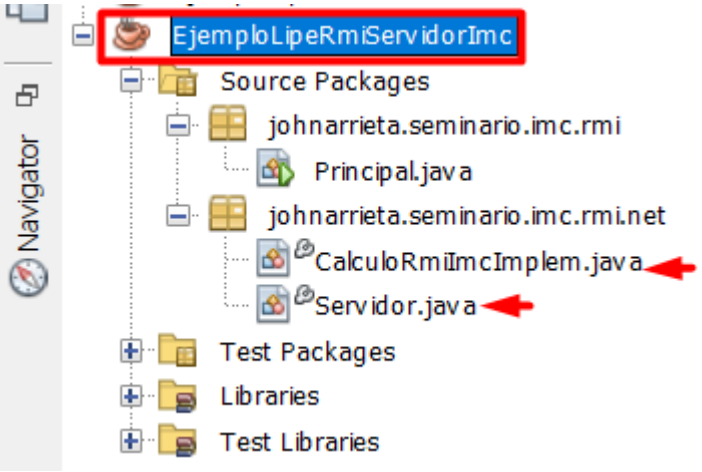


1. En esta nueva ventana “NAME and Location”, colocamos el nombre de la clase.
2. Ponemos la ubicación en el paquete nuevo del proyecto..
3. Damos clic en el botón finish.

4. Crear la clase CalculoRmiImcImplem

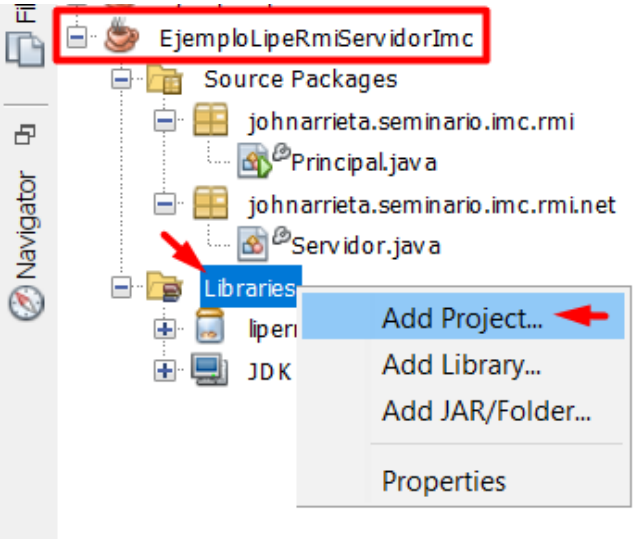


1. Damos clic derecho en el paquete nuevo antes creado, clic en new; seleccionamos java class.
2. En esta nueva ventana “NAME and Location”, colocamos el nombre de la clase.
3. Ponemos la ubicación en el paquete antes seleccionado en el proyecto..
4. Damos clic en el botón finish.

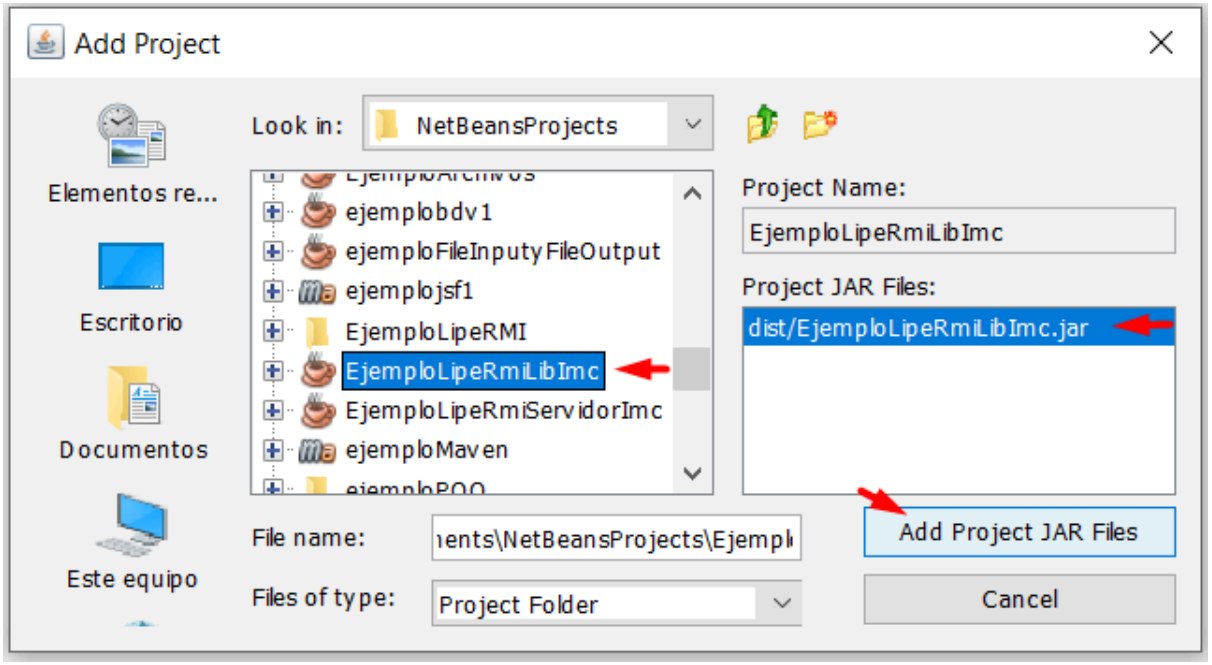


revisamos que las 2 clases se crearon en el mismo paquete.

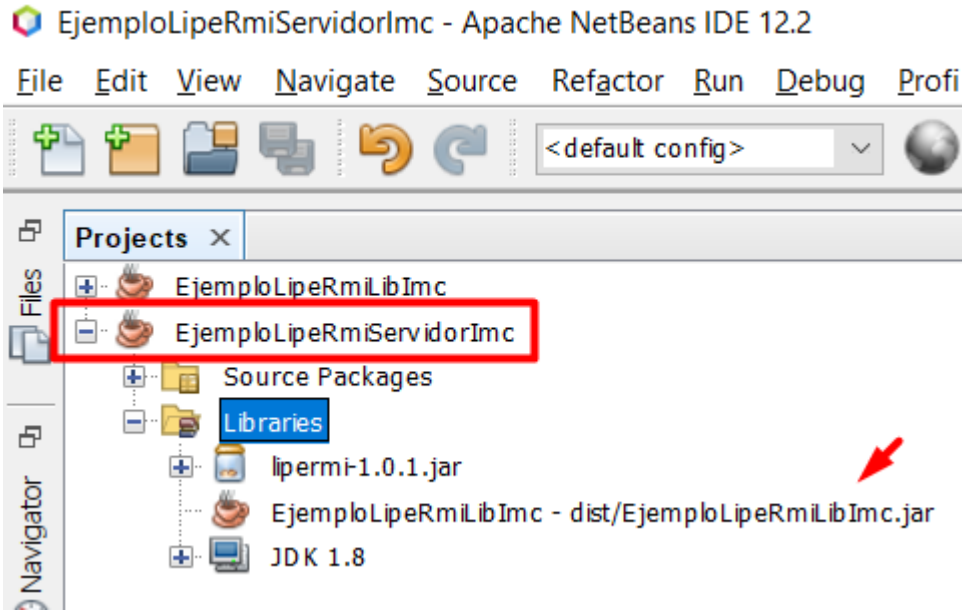
4. Agregar la librería del del proyecto que creamos anteriormente



Hacemos clic derecho en otra vez en la librería del proyecto y accedemos en add Project



Buscamos la primera librería que creamos con anterioridad, seleccionamos y le damos clic en el .jar, finalizamos dando clic en el botón add project JAR FILES



verificamos que el proyecto .jar se agregó con éxito.

5. Escribir el código de las clases.

Clase CalculoRmiImcImplem

```java
package johnarrieta.seminario.imc.rmi.net;

import johnarrieta.seminario.imc.rmi.lib.DatosImc;
import johnarrieta.seminario.imc.rmi.lib.IRemotaCalculoImc;

/**
 *
 * @author JOHN CARLOS ARRIETA ARRIETA
 */
public class CalculoRmiImcImplem implements IRemotaCalculoImc {

    private DatosImc datos;

    public CalculoRmiImcImplem() {
    }

    /*
    public DatosImc calcularImc() {

     */
    @Override
    public DatosImc calcularImc(DatosImc datos) {

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

Primero importamos DatosImc, además IremotacalculoImc, y al lado del nombre de la clase ponemos “implements IremotacalculoImc”.

Creamos un atributo llamado datos de tipo DatosImc. Le agregamos el constructor a la clase, debajo agregamos un método llamado calcularImc, el cual recibe un parámetro de tipo DatosImc.

Dentro del método se crea una variable float llamada resultado, primero hay un if con un else; el if comprueba si los datos enviados son iguales a cero si es asi manda un error, el else tiene la fórmula imc y da un resultado, ese resultado es comparado para ver en qué categoría cabe si flaco, normal y gordo, al final devuelve un resultado de tipo datosimc.

Código de la clase Servidor

```
package johnarrieta.seminario.imc.rmi.net;

import java.io.IOException;
import johnarrieta.seminario.imc.rmi.lib.IRemotaCalculoImc;
import net.sf.lipermi.exception.LipeRMIException;
import net.sf.lipermi.handler.CallHandler;
import net.sf.lipermi.net.Server;

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

    private int puerto = 9007;
    private CallHandler invocador;
    private Server servidor;
    private CalculoRmiImcImplem calculoImc;
    private IRemotaCalculoImc calculoImcRemoto;

    public Servidor() {
        invocador = new CallHandler();
        servidor = new Server();
        calculoImc = new CalculoRmiImcImplem();
    }

    public void iniciar()  throws Exception {
        try {
            invocador.registerGlobal(IRemotaCalculoImc.class, calculoImc);
            servidor.bind(puerto, invocador);
        } catch (LipeRMIException ex) {
            throw new Exception("Error: No es posible invocar metodos remotos");
        } catch (IOException ex) {
             throw new Exception("Error: I/O");
        }
    }

    public void detener() {
        servidor.close();
    }
}
```

se abre la clase servidor. Se importa la clase IOException, IRemotaCalculoImc, lipeRMIException, callHandler y server de la librería lipermi.

Se crean 5 atributos: un puerto, invocador, servidor, calculoImc y calculoImcRemoto. Se agrega el constructor de la clase y en el constructor se instancia 3 atributos invocador, servidor y calculoImc.
Hay también un método llamado inicio que devuelve nada o void, al lado tiene throws Exception, aqui es este método se inicia el servidor y además tiene un try si algo sale mal.
Al final hay un método detener que devuelve vacío y que cierra el servidor.

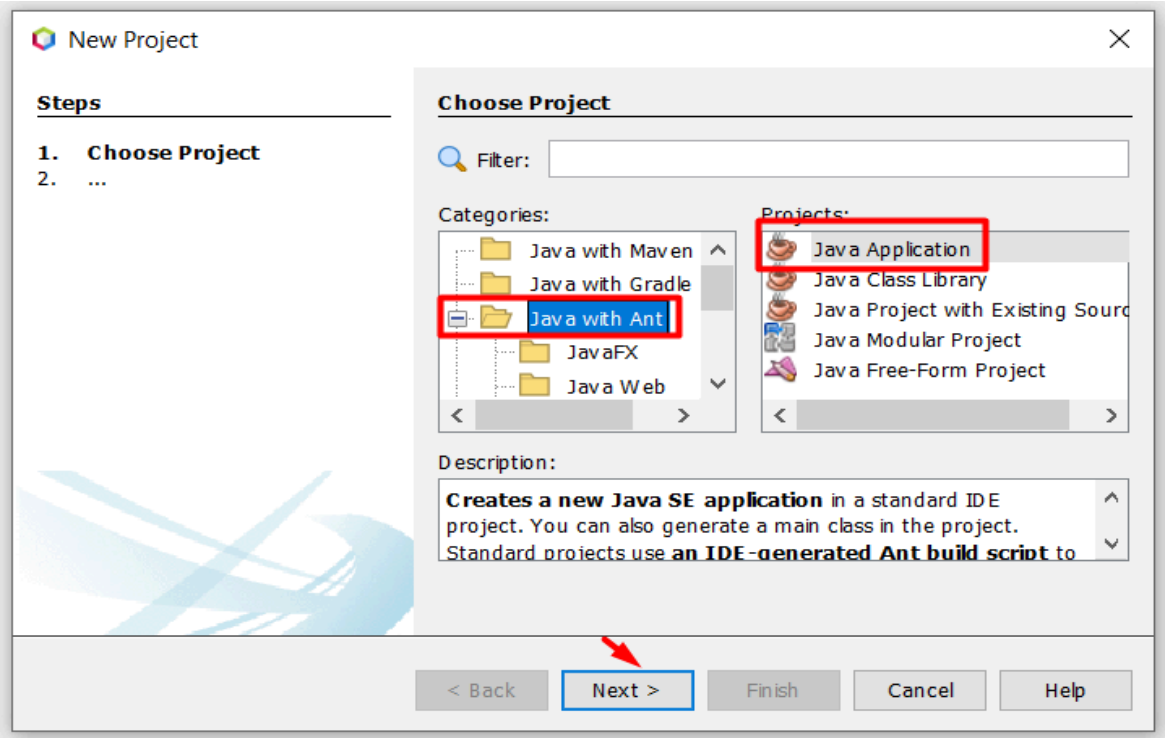
Código de la clase Principal

```
package johnarrieta.seminario.imc.rmi;

import johnarrieta.seminario.imc.rmi.net.Servidor;

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

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        Servidor servicio = new Servidor();
        try {
            servicio.iniciar();
        } catch (Exception ex) {
            System.out.println(ex.getLocalizedMessage());
        }
    }

}
```

Se crea en la main class la instancia de la ventana principal, y se muestra.

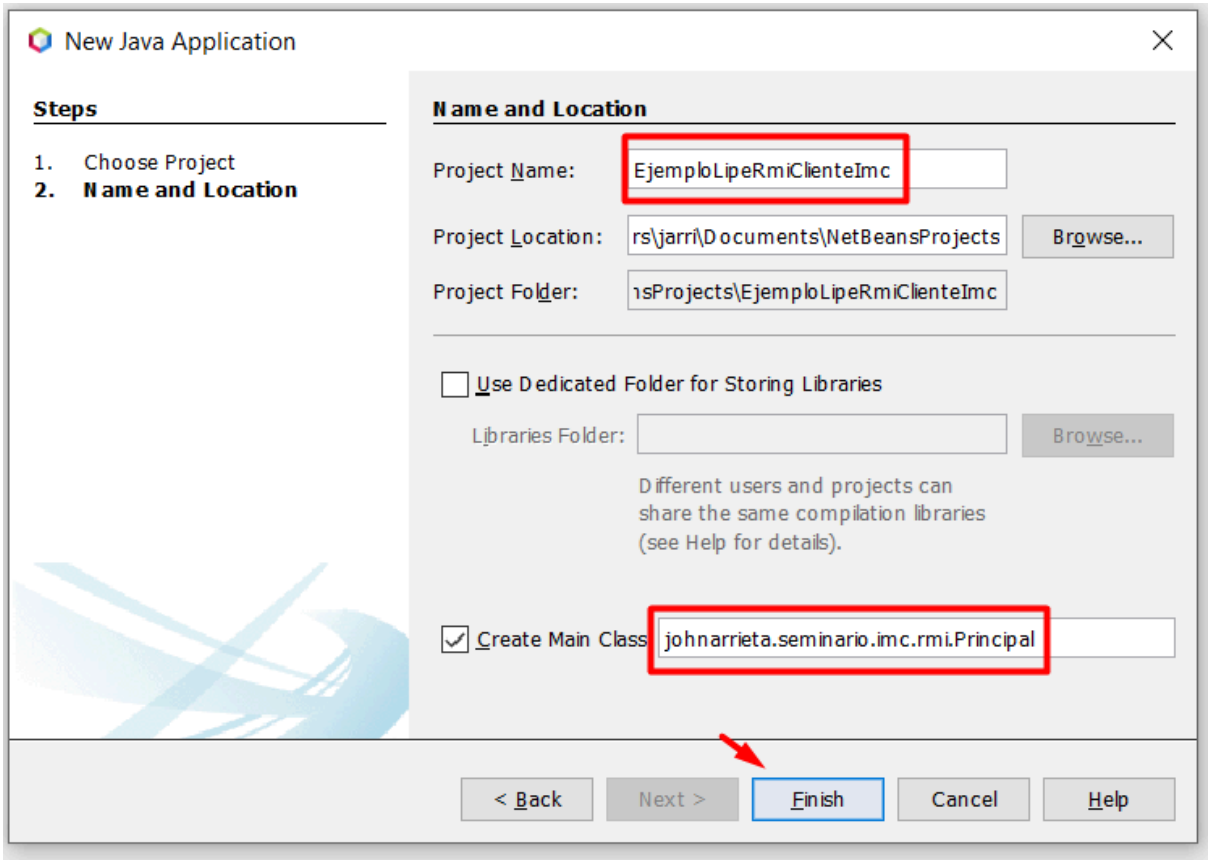
CREAR EL PROYECTO PARA LA APLICACIÓN CLIENTE

1. Crear un nuevo proyecto de tipo Java Application



Lo primero es crear el nuevo archivo para el cliente :

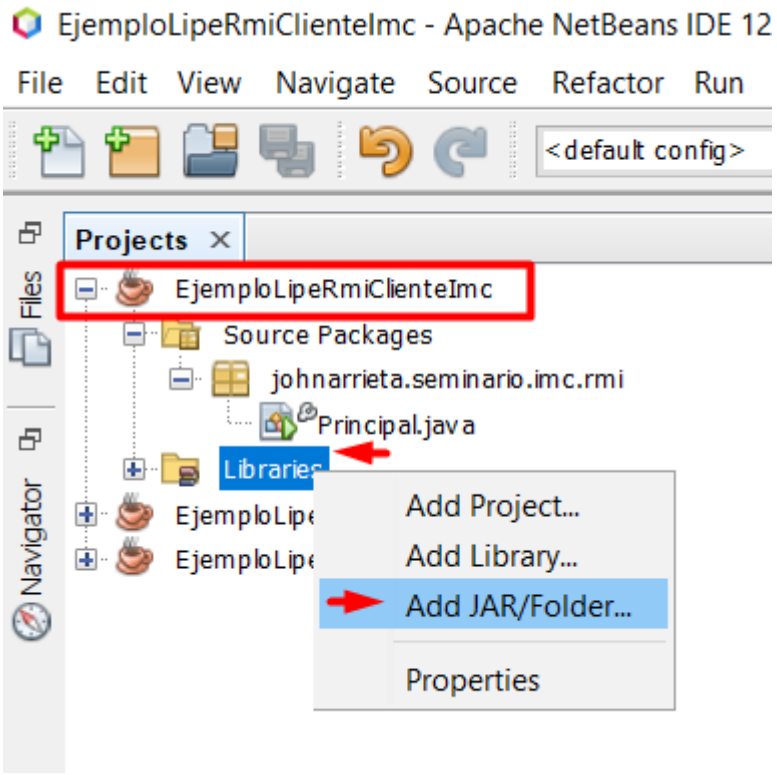
1. Dar click en el botón nuevo proyecto.
2. Escoger la categoría java with ant.
3. y después elegir Java Application.
4. por último dar click en el botón next.



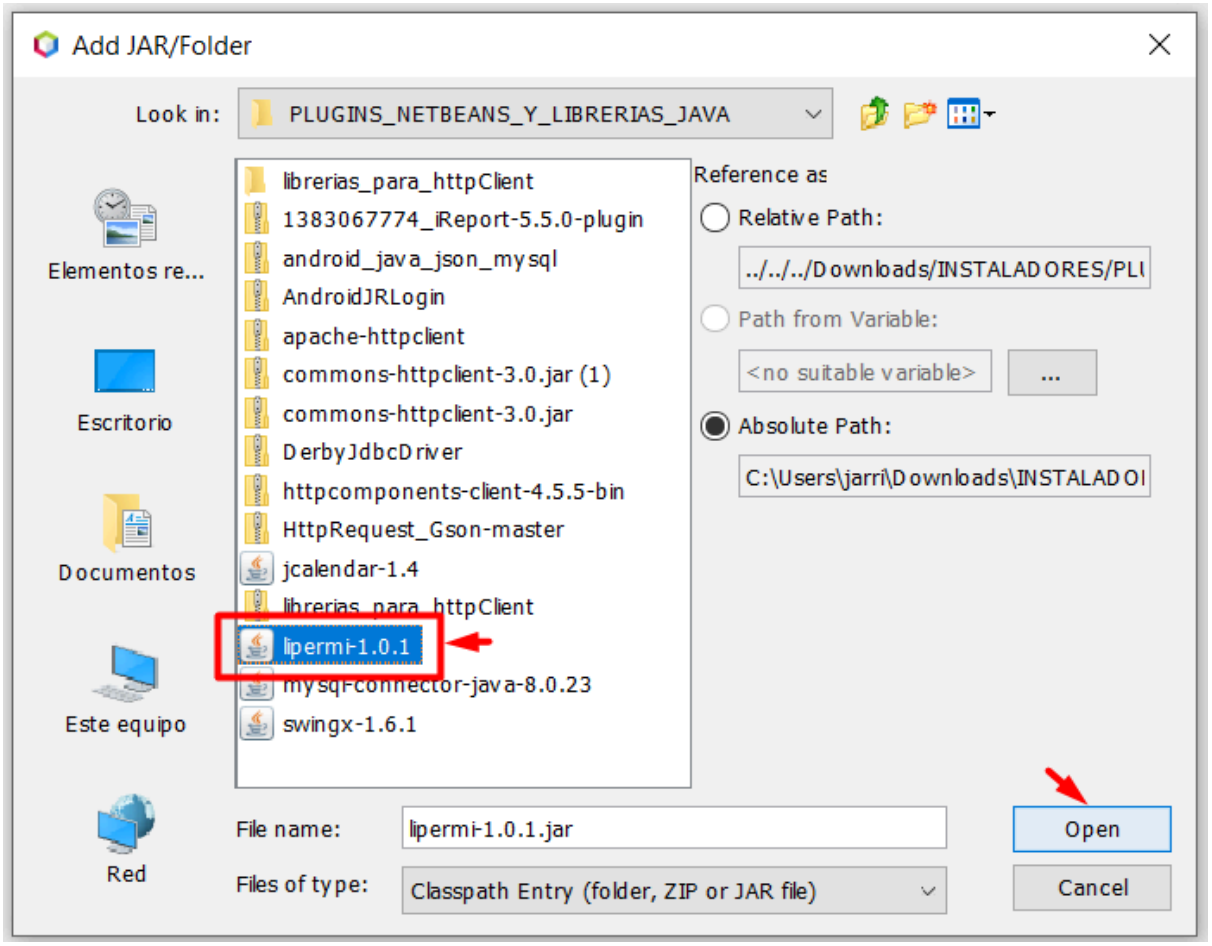
En esta ventana le ponemos nombre al proyecto esta es la parte del cliente:

1. En la casilla de project Names; colocamos el nombre del proyecto
2. damos click en el botón Browse
3. encontramos la carpeta donde queremos guardar el proyecto.
4. dar click en la casilla cuadrada y
5. crear la clase principal..
6. finalizamos todo dando click en el botón finish

2. Agregar la librería LipeRMI

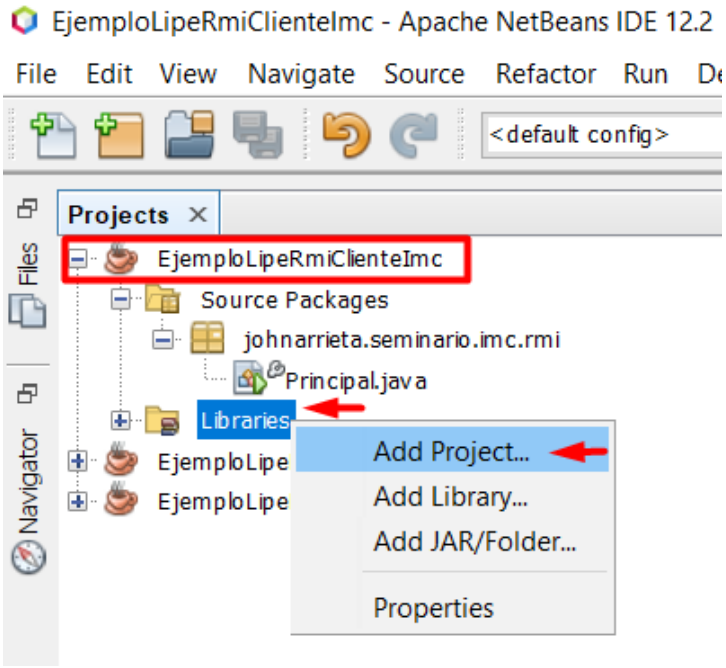


Ahora agregamos la librería LipeRMI, damos clic derecho en la carpeta librerías; después click en la opción add JAR/Folder.

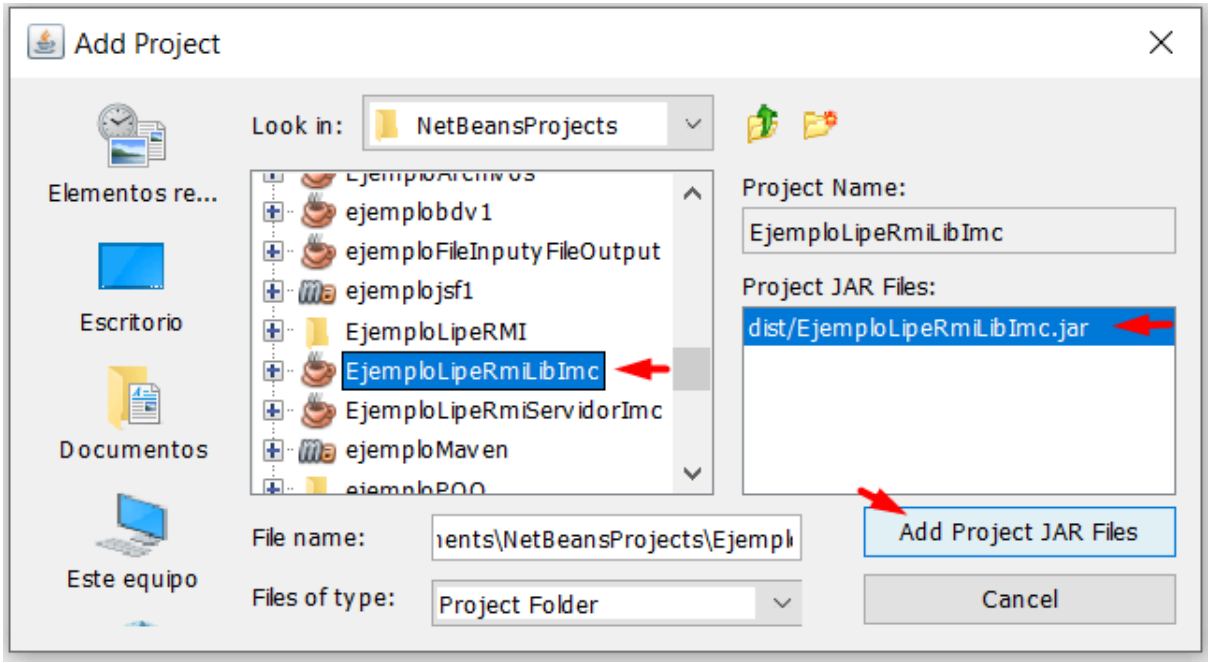


en la ventana add JAR/Folder; agregamos la librería lipermi, dando clic en el archivo y después click en el botón open.

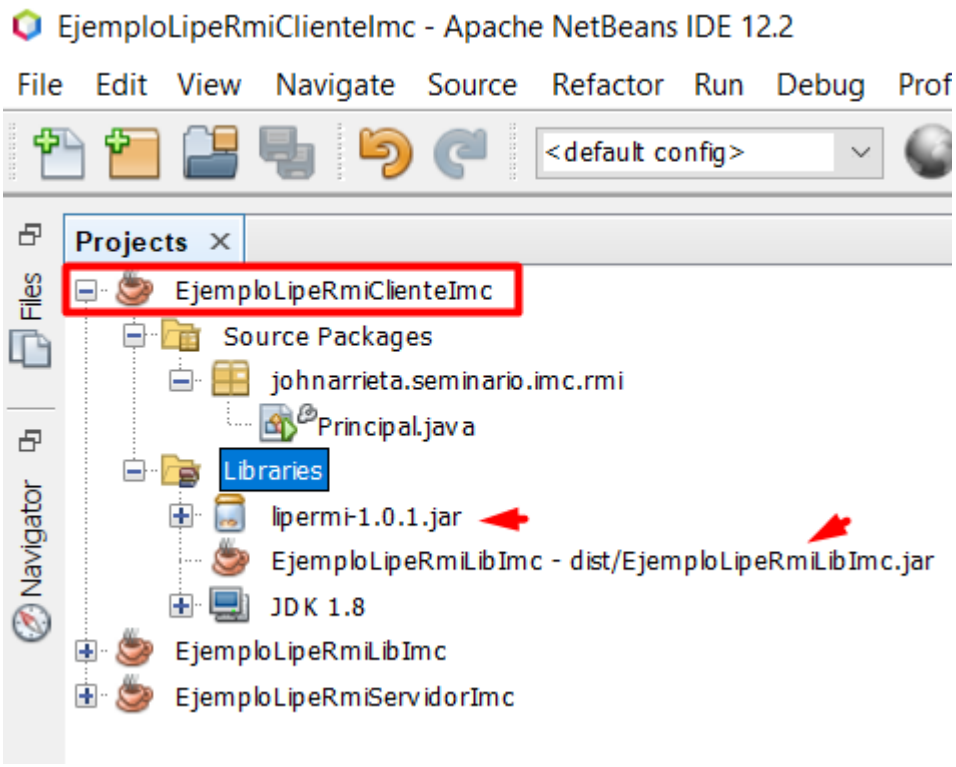
3. Agregar la librería del proyecto creado inicialmente



Clic derecho en librerías del proyecto cliente, clic en add project

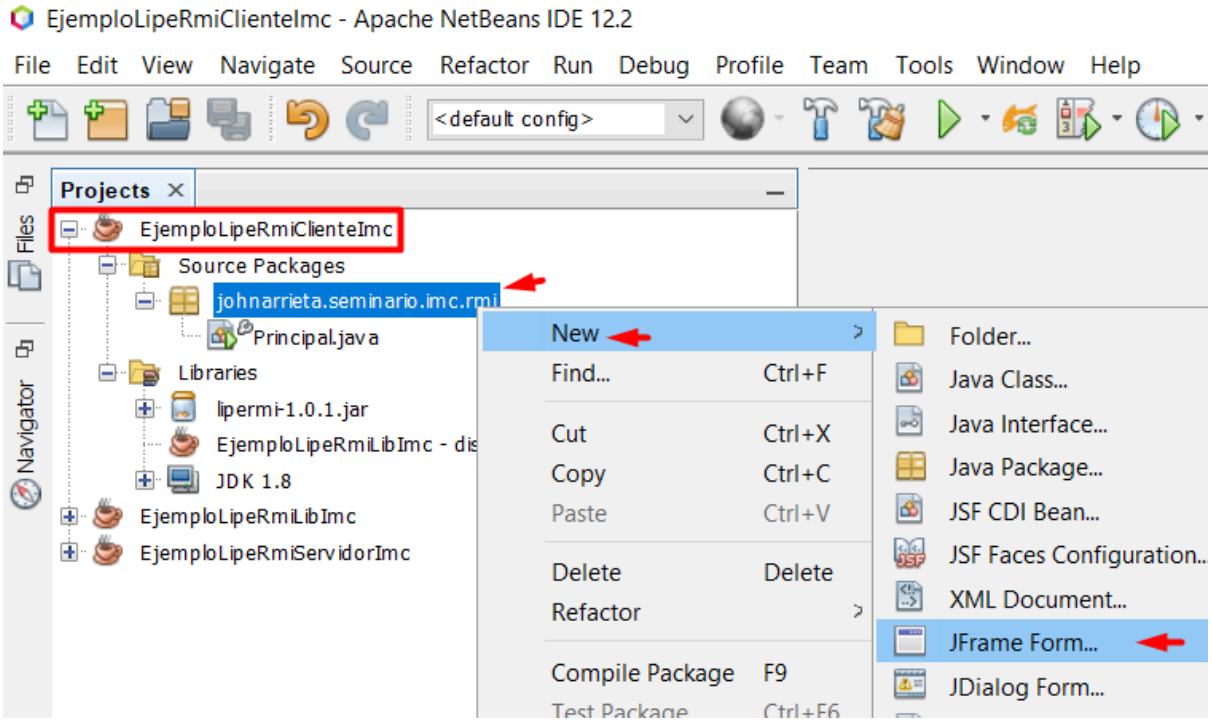


Buscamos la primera librería que creamos con anterioridad, seleccionamos y le damos clic en el .jar, finalizamos dando clic en el botón add project JAR FILES

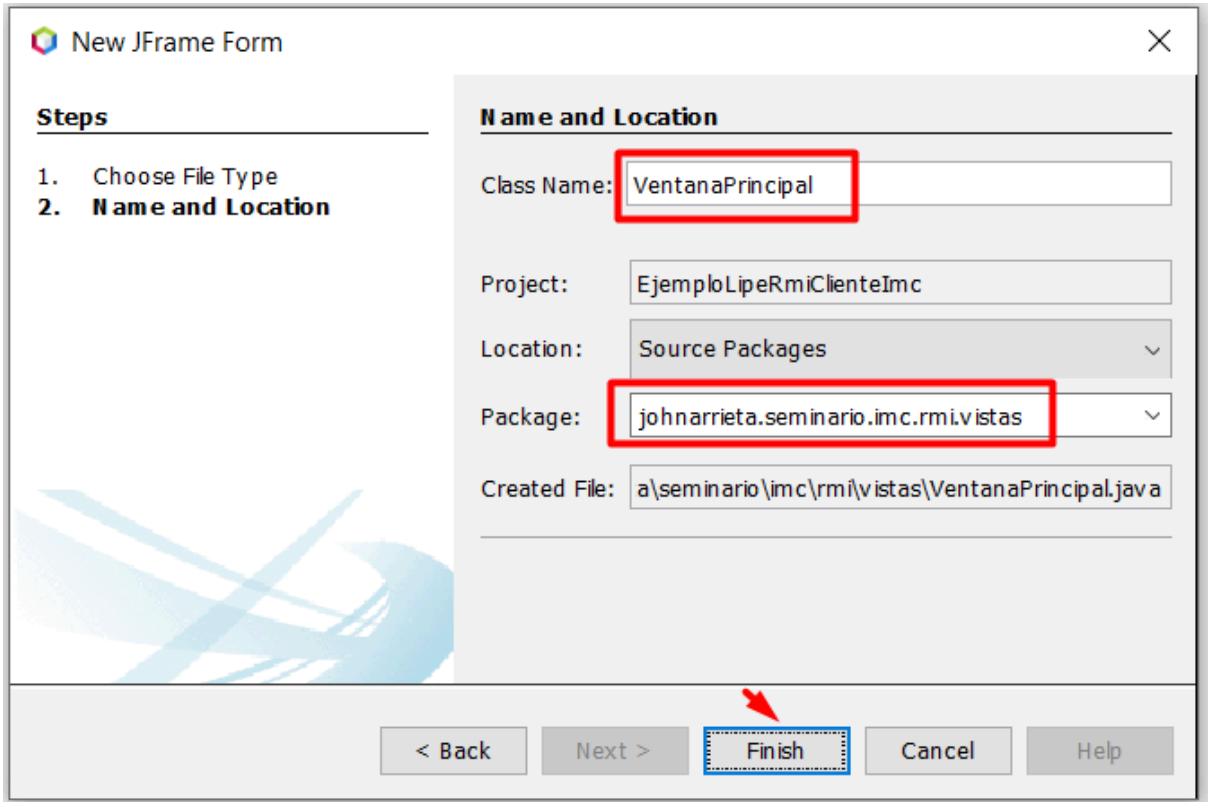


verificamos que se agregó bien todo.

4. Crear la clase Ventana Principal.



Damos click derecho en un paquete, saldrá un menú con diferente opciones y damos click en new y después click en jFrame Form.

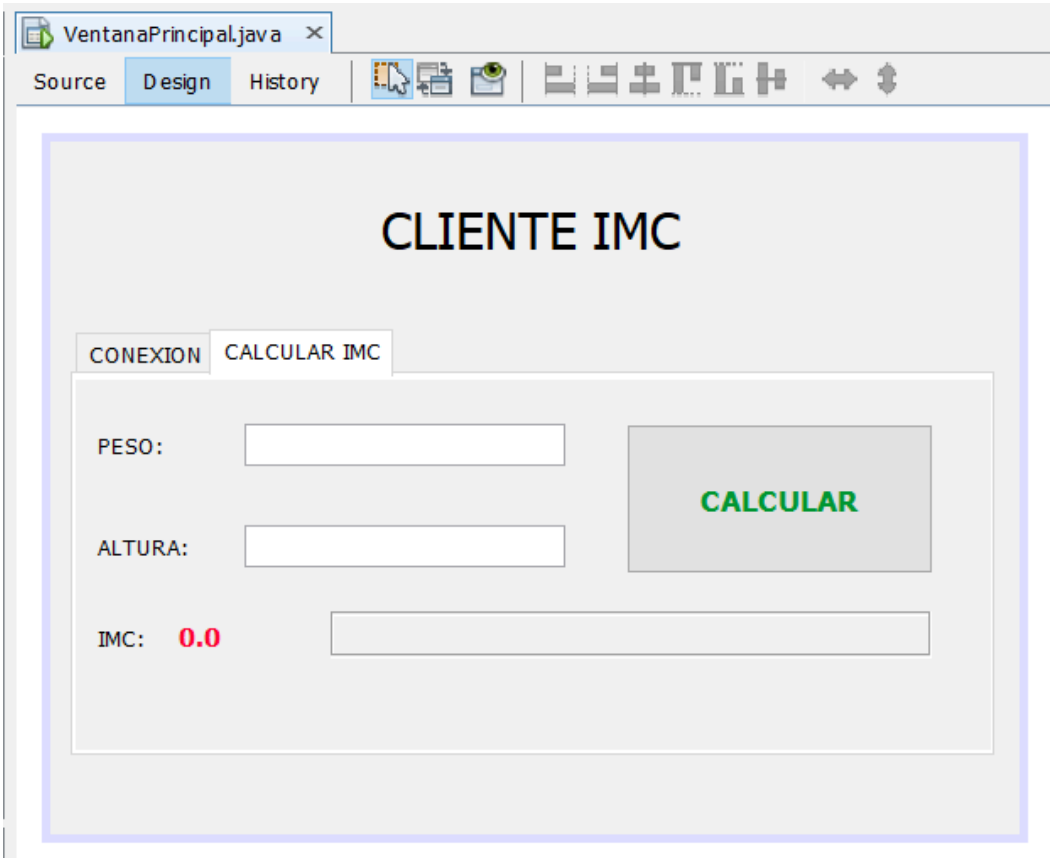


En esta ventana le ponemos nombre al proyecto esta es la parte del “VentanaPrincipal”

1. En la casilla de project Names; colocamos el nombre del proyecto” VentanaPrincipal”
2. En la casilla de package colocamos el nombre del paquete nuevo que se va crear.
3. finalizar todo, dando click en el botón finish

5. Diseñar el formulario y programar los eventos de la GUI.

En el archivo ventana principal del cliente, creamos la ventana de la parte del cliente con un jtable creamos el nombre “cliente IMC”, debajo ponemos un tabbedpane; dentro de él ponemos 2 panel, en el primer panel llamado conexión ponemos dos textfield; uno para dirección ip y otro para el puerto (cada textfield tiene su respectivo jlabel) además dos jlabel; uno para el estado y otro para la respuesta del estado, al final con un botón con el nombre conectar.



en el otro panel lo llamamos calcular imc, dentro ponemos un tabbedpane, en el le ponemos dos textfield; uno para peso y otro para el altura (cada textfield tiene su respectivo jlabel con su nombre) además dos jlabel; uno que diga IMC y otro para la respuesta del IMC, al final con un botón con el nombre calcular.

Código de los dos principales manejadores (Oyentes de Eventos) más importantes de esta GUI

```
    private void btnIniciarActionPerformed(java.awt.event.ActionEvent evt) {
228
229         try {
230             if (btnIniciar.getText().equalsIgnoreCase("Conectar")) {
231
232                 puerto = Integer.parseInt(campoPuertoServidor.getText());
233                 ipServidor = campoIPServidor.getText();
234                 invocadorRemoto = new CallHandler();
235                 cliente = new Client(ipServidor, puerto, invocadorRemoto);
236                  calculoImcRemoto =  (IRemotaCalculoImc) cliente.getGlobal(IRemotaCalculoImc.class);
237                 btnIniciar.setText("Desconectar");
238                 btnIniciar.setForeground(Color.RED);
239                 txtEstado.setText("Conectado");
240                 txtEstado.setForeground(Color.GREEN);
241             } else if (btnIniciar.getText().equalsIgnoreCase("Desconectar")) {
242                 cliente.close();
243                 btnIniciar.setText("Conectar");
244                 txtEstado.setText("Desconectado");
245                 btnIniciar.setForeground(Color.GREEN);
246                 txtEstado.setForeground(Color.RED);
247             }
248         } catch (IOException ex) {
249             System.out.println("ERROR AL CONECTAR");
              ex.printStackTrace();
251         }
252     }
```

Se importan todas las clases que se ven en la imagen. Se crea una variable de tipo string, int, client, IRemotaCalculoImc y callHandler.

En el actionperformed del botón iniciar se crea un try catch; en el try; tiene un if que para entrar necesita que el botón iniciar tenga el texto “conectar”, dentro del if en la variable puerto se le asigna el número que tenga el campoPuerto, a la variable ipServidor se le agrega tambien la ip servidor que esta en el campo del formulario, se instancia el callHandler y se agrega a la variable invocador remoto, además se crea una variable tipo cliente y se agregan las variables antes dichas en la instancia, se llama tambien el metodo que calcula el imc, el boton cambia de texto y el label de estado también cambia a conectado, y un else if se activa si el botón tiene en su texto desconectar, aqui el servidor se cierra, el label cambia y el texto del botón cambia. En el catch se agrega como argumento a una variable de tipo IOException además imprime un error e imprime la variable.

```
    private void btnIniciar1ActionPerformed(java.awt.event.ActionEvent evt) {

        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);
                    DatosImc datos = new DatosImc();
                    datos.setAltura(altura);
                    datos.setPeso(peso);
                    System.out.println("Enviados los datos\nEsperando respuesta");
                    datos = calculoImcRemoto.calcularImc(datos);
                    System.out.println("IMC: " + datos.getResultado()+ "\nMensaje: " + datos.getInterpretacion());
                    txtResultado.setText(datos.getResultado() + "");
                    txtMensaje.setText(datos.getInterpretacion());
                } catch (Exception ex) {
                    JOptionPane.showMessageDialog(VentanaPrincipal.this, "ERROR con el cliente " + ex.getMessage());
                    System.out.println("ERROR con el cliente " + ex.getMessage());
                    ex.printStackTrace();
                }
            }
        };
        hilo.start();
    }
```

En el actionperformed del botón iniciar1 se crean dos variables peso y altura que obtienen la información de los campos del formulario, se crea un hilo o thread A continuación, el programa convierte el texto de entrada en variables de tipo float, lo que permite realizar operaciones matemáticas con estos valores. Se crea y ejecuta un nuevo hilo para realizar tareas en segundo plano. Este enfoque permite que el programa principal siga funcionando sin interrupciones, mejorando la eficiencia y la capacidad de respuesta. El programa también maneja la impresión y escritura de variables en la consola o en una salida externa. Esto es crucial para la visualización y registro de datos, así como para la interacción con otros sistemas o componentes. Todo esta información la agrega al objeto datos que es tipo DatosImc. Envía la información aal metodo calcular imc, y regresa un resultado. Finalmente, el programa incluye mecanismos para leer valores y manejar posibles errores de entrada/salida.

Código completo de la clase Ventana Principal

```java
package johnarrieta.seminario.imc.rmi.vistas;

import java.awt.Color;
import java.io.IOException;
import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JOptionPane;

import johnarrieta.seminario.imc.rmi.lib.*;

import net.sf.lipermi.handler.CallHandler;
import net.sf.lipermi.net.Client;

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

    /**
     * Creates new form VentanaPrincipal
     */
    CallHandler invocadorRemoto;
    String ipServidor = "localhost";
    int puerto = 9007;
    IRemotaCalculoImc calculoImcRemoto;
    Client cliente;

    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();
        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("DIRECCIÓN 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()
                .addContainerGap()

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

.addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                    .addComponent(jLabel4)
                    .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.LEADING)
                    .addGroup(jPanel3Layout.createSequentialGroup()
                        .addComponent(jLabel7)
```

```
.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(jLabel5, 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)

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

.addGroup(jPanel3Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
                            .addComponent(jLabel5)
                            .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(jLabel6)
                            .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(jLabel7)
                    .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(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>
```

```
  private void btnIniciarActionPerformed(java.awt.event.ActionEvent evt) {
      try {
          if (btnIniciar.getText().equalsIgnoreCase("Conectar")) {
              puerto = Integer.parseInt(campoPuertoServidor.getText());
              ipServidor = campoIPServidor.getText();
              invocadorRemoto = new CallHandler();
              client = new Client(ipServidor, puerto, invocadorRemoto);
              calculoImcRemoto =  (IRemotaCalculoImc)
              cliente.getGlobal(IRemotaCalculoImc.class);
              btnIniciar.setText("Desconectar");
              btnIniciar.setForeground(Color.RED);
              txtEstado.setText("Conectado");
              txtEstado.setForeground(Color.GREEN);
          } else if (btnIniciar.getText().equalsIgnoreCase("Desconectar")) {
              cliente.close();
              btnIniciar.setText("Conectar");
              txtEstado.setText("Desconectado");
              btnIniciar.setForeground(Color.GREEN);
              txtEstado.setForeground(Color.RED);
          }
      } catch (IOException ex) {
          System.out.println("ERROR AL CONECTAR");
          ex.printStackTrace();
      }
  }
```

```
private void btnIniciar1ActionPerformed(java.awt.event.ActionEvent evt) {
    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);
                DatosImc datos = new DatosImc();
                datos.setAltura(altura);
                datos.setPeso(peso);
                System.out.println("Enviados los datos\nEsperando
                                                                respuesta");
                datos = calculoImcRemoto.calcularImc(datos);
                System.out.println("IMC: " + datos.getResultado()+
                                   "\Mensaje: " + datos.getInterpretacion());
                txtResultado.setText(datos.getResultado() + "");
                txtMensaje.setText(datos.getInterpretacion());
            } catch (Exception 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;
 }

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

```
// 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
}
```

Se importan todas las clases que se ven en la imagen. Se crea una variable de tipo string, int, client, IRemotaCalculoImc y callHandler.

En el actionperformed del botón iniciar se crea un try catch; en el try; tiene un if que para entrar necesita que el botón iniciar tenga el texto “conectar”, dentro del if en la variable puerto se le asigna el número que tenga el campoPuerto, a la variable ipServidor se le agrega tambien la ip servidor que esta en el campo del formulario, se instancia el callHandler y se agrega a la variable invocador remoto, además se crea una variable tipo cliente y se agregan las variables antes dichas en la instancia, se llama tambien el metodo que calcula el imc, el boton cambia de texto y el label de estado también cambia a conectado, y un else if se activa si el botón tiene en su texto desconectar, aqui el servidor se cierra, el label cambia y el texto del botón cambia. En el catch se agrega como argumento a una variable de tipo IOException además imprime un error e imprime la variable.

En el actionperformed del botón iniciar1 se crean dos variables peso y altura que obtienen la información de los campos del formulario, se crea un hilo o thread A continuación, el programa convierte el texto de entrada en variables de tipo float, lo que permite realizar operaciones matemáticas con estos valores. Se crea y ejecuta un nuevo hilo para realizar tareas en segundo plano. Este enfoque permite que el programa principal siga funcionando sin interrupciones, mejorando la eficiencia y la capacidad de respuesta. El programa también maneja la impresión y escritura de variables en la consola o en una salida externa. Esto es crucial para la visualización y registro de datos, así como para la interacción con otros sistemas o componentes. Todo esta información la agrega al objeto datos que es tipo DatosImc. Envía la información aal metodo calcular imc, y regresa un resultado. Finalmente, el programa incluye mecanismos para leer valores y manejar posibles errores de entrada/salida.

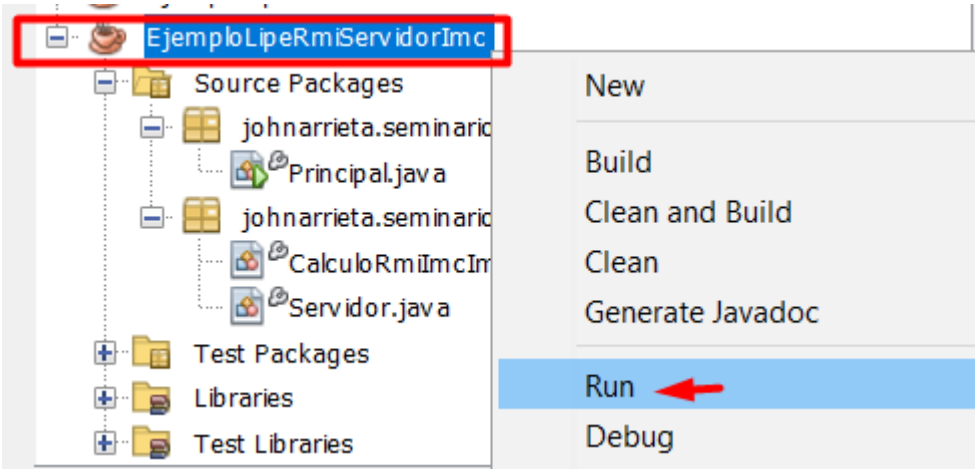
Código de la clase Principal

```java
package johnarrieta.seminario.imc.rmi;

import johnarrieta.seminario.imc.rmi.vistas.VentanaPrincipal;

/**
 *
 * @author jarri
 */
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);
    }
}
```

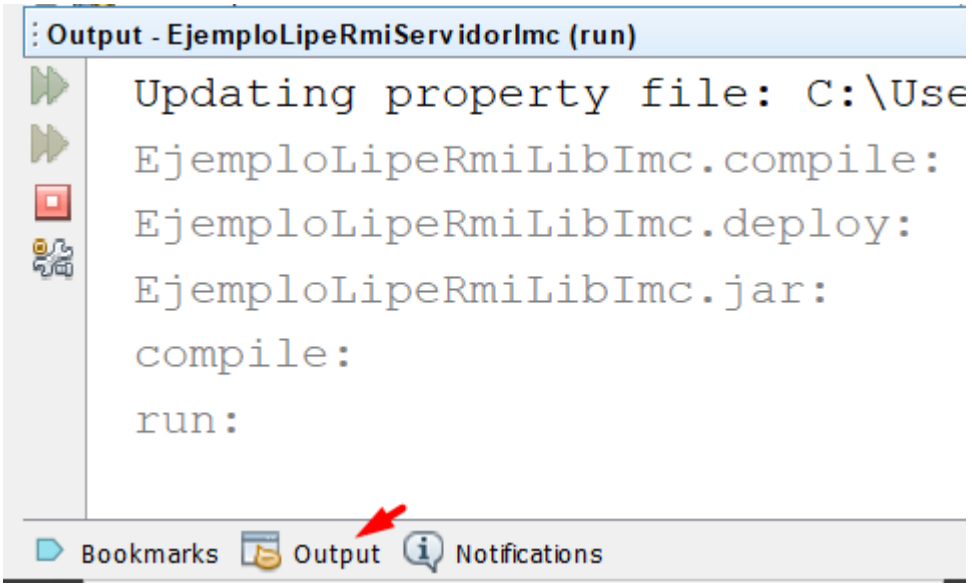
Se crea en la main class la instancia de la ventana principal, y se muestra.

Probar el sistemas distribuido cliente servidor

1) Ejecutar la aplicación Servidor

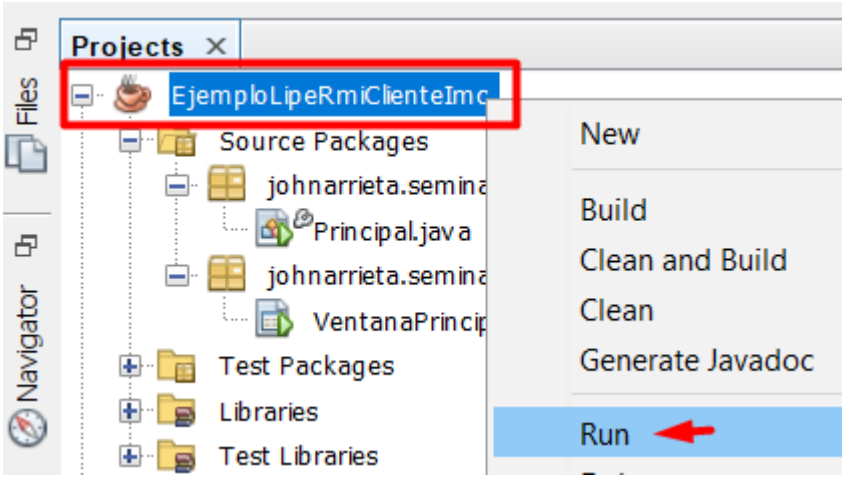


primero se ejecuta el servidor

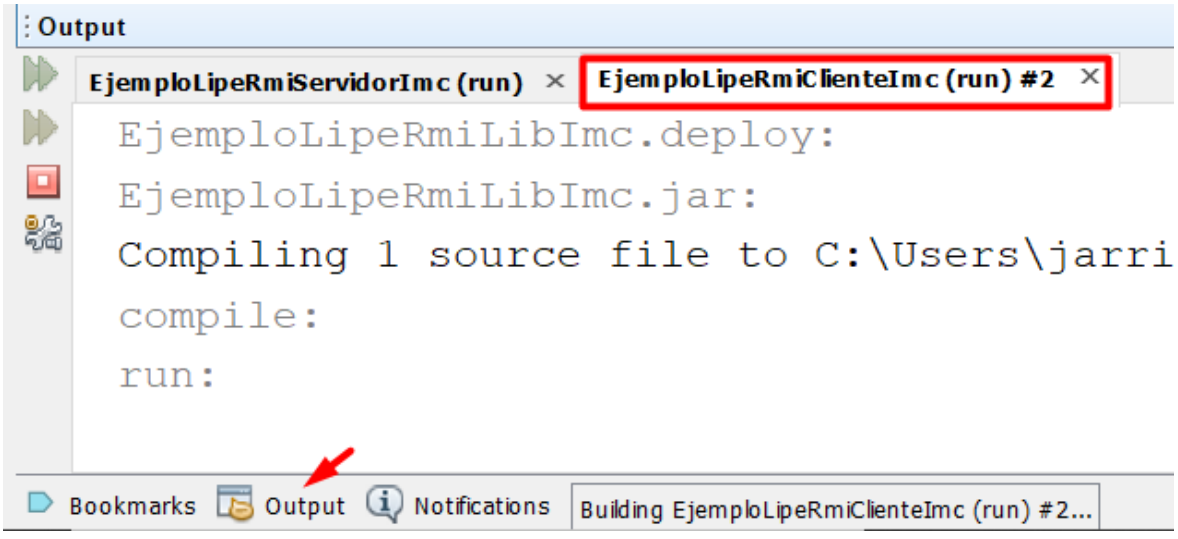


En output se puede ver que es lo que está pasando con el servidor si hay algún error o todo está bien.

2) Ejecutar la aplicación Cliente

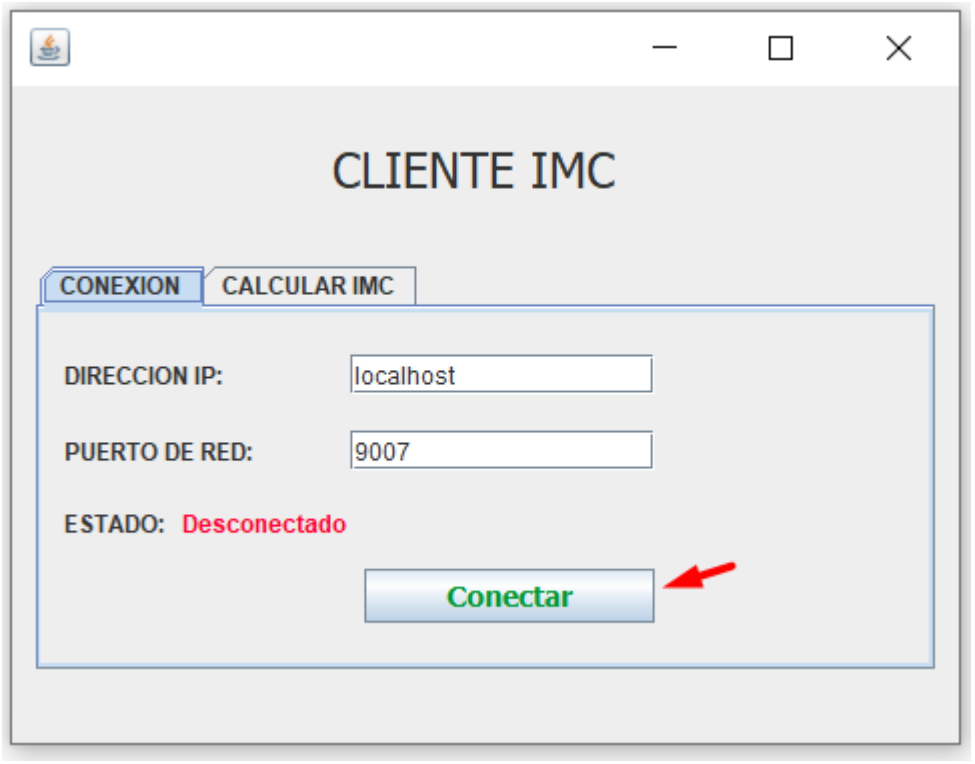


Se ejecuta el cliente.



En output se puede ver que es lo que está pasando con el Cliente si hay algún error o todo está bien.

3) Iniciar la conexión desde el cliente hacia el servidor



hay que probar la aplicación se coloca el la dirección ip del servidor y un puerto, se da al botón conectar

4) Ingresar datos, solicitar realizar el cálculo de forma remota y mostrar el resultado.