

Colegio: Centro Educativo Técnico laboral KINAL

Profesor: Jorge Luis Pérez Canto

Código Técnico: IN5BV

Carné: 2020335

# Tarea de la calculadora

Nombre: Carlos Adolfo Alvarez Crúz

Ciudad de Guatemala 30 de abril de 2021



---

# MANUAL CALCULADORA

---

Indicaciones de uso



30 DE ABRIL DE 2021  
CENTRO EDUCATIVO TECNICO LABORAL KINAL  
IN5BV

# Contenido

**INTRODUCCIÓN:..... 3**

**REQUISITOS FUNCIONALES:..... 4**

**REQUISITOS DE SOFTWARE Y HADWARE:..... 6**

**IMPLEMENTACIÓN: ..... 7**

**DESCRIPCIÓN DE LOS ALGORITMOS: ..... 15**

**DISEÑO:..... 29**

## INTRODUCCIÓN:

Son muchas las funciones que le podemos dar a un lenguaje que es orientado a objetos, pues esta vez toca ver como podemos desarrollar una calculadora con el lenguaje de programación de java en el transcurso de este manual podrá observar que funcionalidad tiene desarrollar una calculadora en java como por ejemplo sumar, restar, multiplicar y dividir dos o mas números, también conoceremos cuales son los requisitos para que nuestra computadora pueda ejecutar sin problemas la calculadora, una explicación mas afondo de como ejecutar y usar la calculadora, una explicación del código para darle vida a la calculadora y por último que s desarrollo la interfaz gráfica ósea el diseño de la calculadora espero que esta instructivo sea de su agrado

## REQUISITOS FUNCIONALES:

Las funcionalidades que la calculadora posee son muy simples, ya que con ella podemos sumar dos o mas números, podemos restar dos o mas números, multiplicar dos o más número, dividir dos o más números e incluso podemos sacar el promedio de un número, podemos encontrar el positivo y negativo incluso podemos hacer operaciones con decimales.

AC		%	/
7	8	9	*
4	5	6	-
1	2	3	+
+/-	0	.	=

Algunas de las funcionalidades que le podemos dar a esta calculadora es para hacer cálculos con números muy grandes

$$\begin{aligned} 36 &= 20 + 30 + 2 + 6 = \\ &= 50 + 8 = 58 \\ +41 &= 30 + 40 + 5 + 1 = \\ &= 70 + 6 = 76 \end{aligned}$$

Para estar seguros si los cálculos que hicimos están bien

$$\begin{array}{r}
 1234 \\
 \times 56789 \\
 \hline
 11106 \\
 9872 \\
 8638 \\
 7404 \\
 6170 \\
 \hline
 70077626
 \end{array}$$

$$\begin{array}{|c|c|}
 \hline
 1234 & 56789 \\
 \hline
 \end{array}
 *
 =$$

$$7.0077624E7$$

Otra forma de su utilidad es el ahorro de tiempo al hacer las operaciones



Entre otras cosas...

## REQUISITOS DE SOFTWARE Y HADWARE:

Los requisitos mínimos que necesitamos para que la calculadora corra son los siguientes:

- Un Pentium 2 a 266MHz
- RAM: 4gb
- Espacio en disco 250gb
- Sistema operativo Windows Vista
- Java 11
- JDK 11
- JavaFX 11
- NetBeans 8

Con estos requisitos experimentamos algunos problemas a la hora de ejecutar, por lo que no son muy recomendados, aquí también se puede utilizar el java 8.1 y NetBeans versión 8 pero por problemas de compatibilidad no es recomendable

Para que el programa corra bien y sin ningún error serían los siguientes:

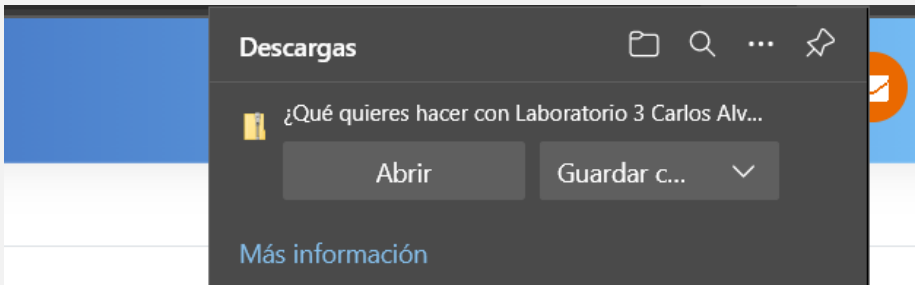
- Un CORE I3 10TH GEN
- RAM: 8gb
- Espacio es disco: 250gb
- Sistema operativo Windows 10
- Java 11
- JDK 11
- JavaFX 11
- NetBeans 12

Con estos requisitos no ocurrirá ningún error y no tendremos problemas al ejecutar la calculadora

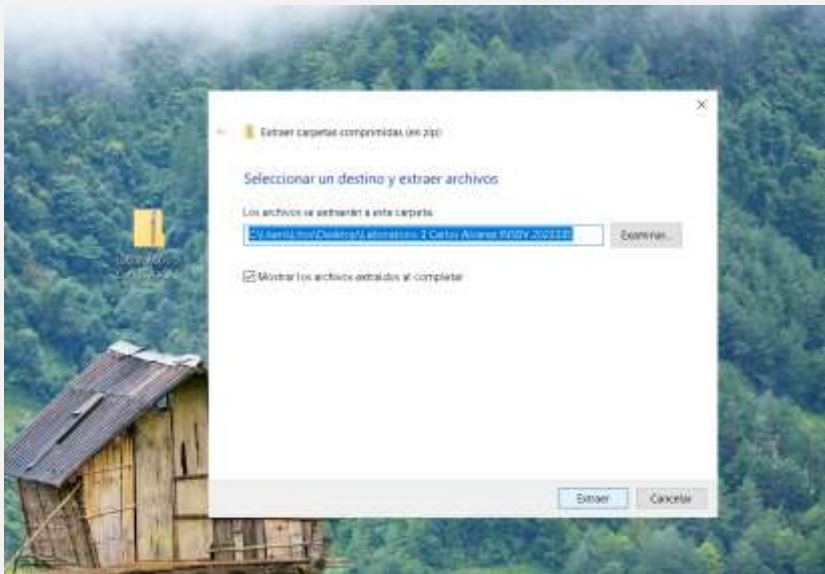
## IMPLEMENTACIÓN:

Los pasos para poder usar esta calculadora son los siguientes

1. Tenemos que tener descargar el proyecto de Academy Kinal



2. Al tenerlo descargado lo descomprimos

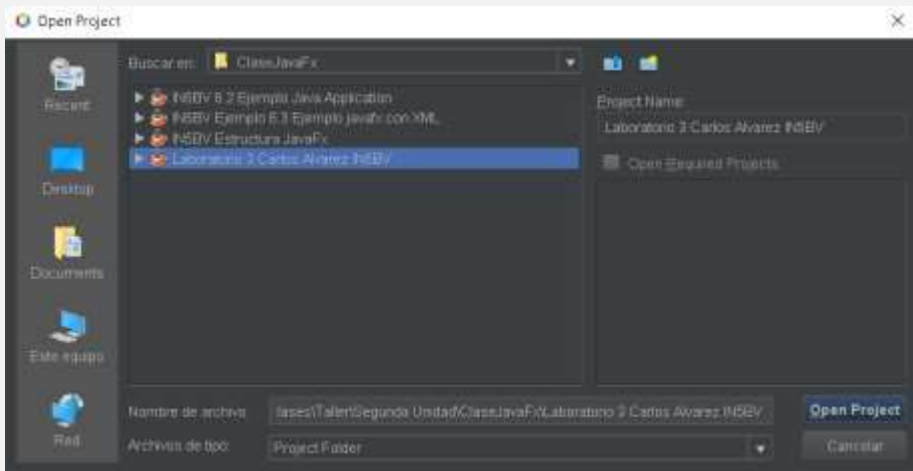




3. Abrimos NetBeans y nos dirigimos a la parte de abrir proyecto




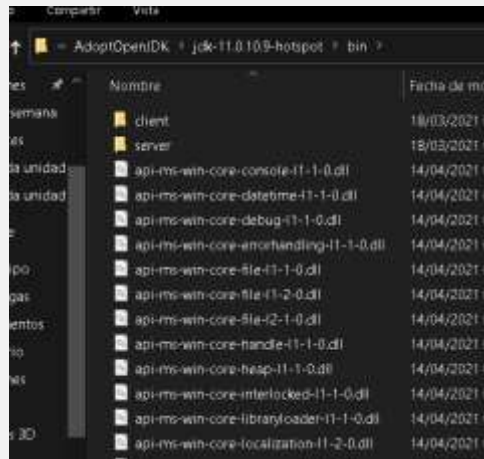
4. Lo buscamos donde está la carpeta descomprimida y le damos abrir



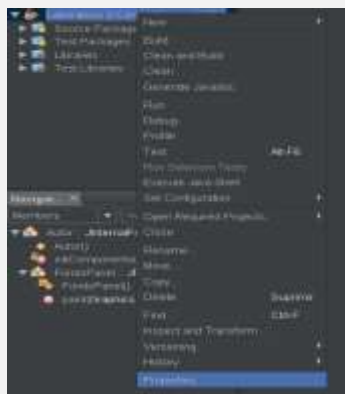
5. Si tienes java 8 puedes saltarte los pasos y dirigirte al paso 6, siguientes si tienes java 11 tendremos que hacer los siguientes pasos

- Tener descargado el JavaFX 11 y configurado correctamente

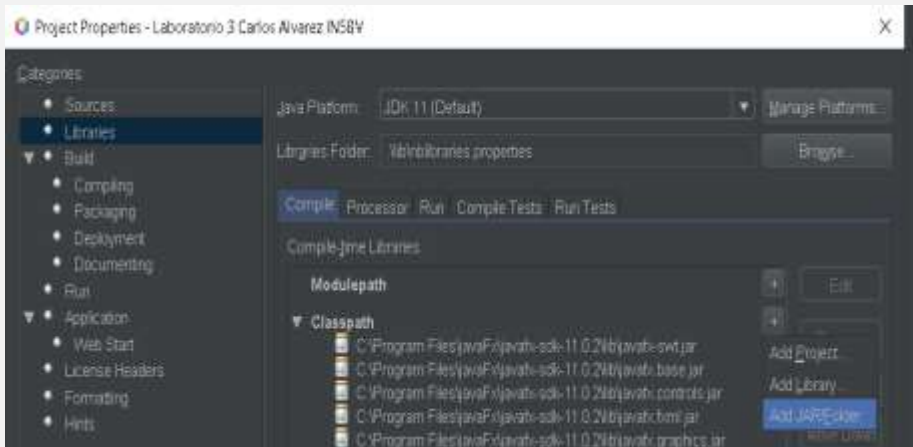
 openjfx-11.0.2\_windows-x64\_bin-sdk



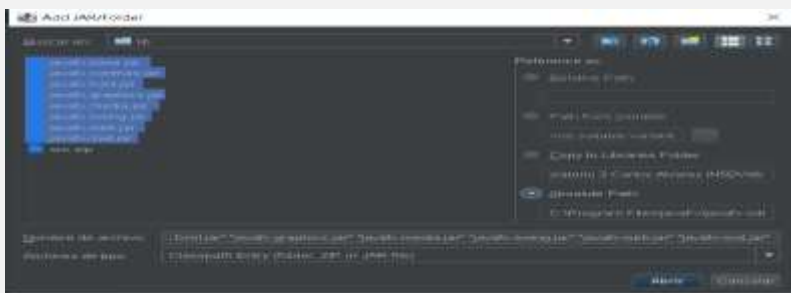
- Al tener abierto el proyecto le damos clic derecho al nombre del proyecto y propiedades ya que tendremos que exportar las librerías de JavaFX ya que el java 11 no lo trae instalado



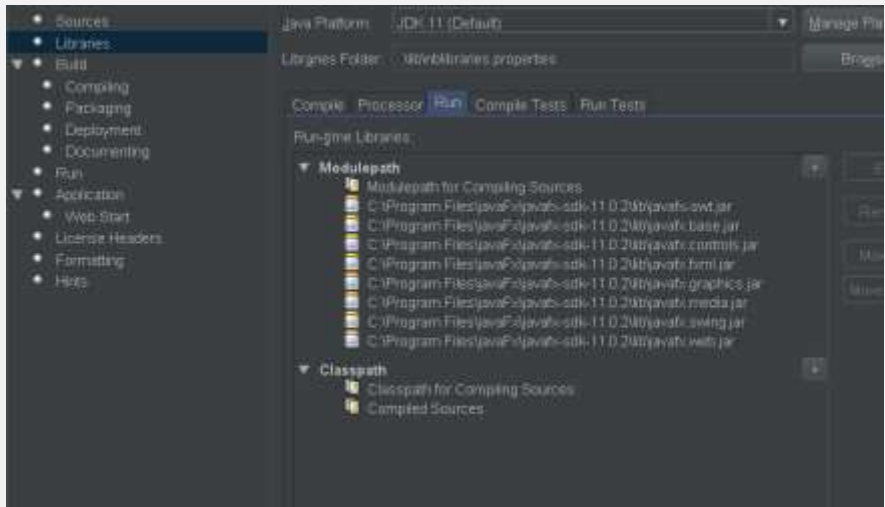
- Nos dirigimos al apartado de “Libraris” y a donde dice “Compile” y al apartado de “Classpath” ahí le damos a la flechita mas y seleccionamos la tercera opción



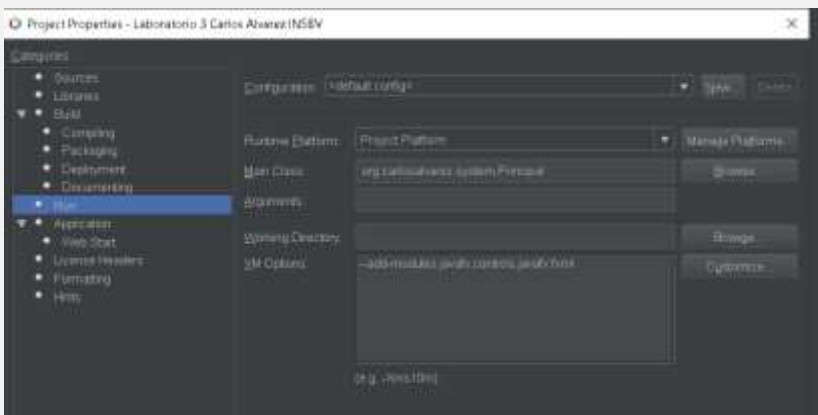
- A l tenerlo abierto buscamos donde hemos instalado JavaFX y seleccionamos todos los .JAR y le damos donde dice abrir, OJO tenemos que tener activada lo opción donde dice Absolute Path ya que ahí NetBeans buscara la ruta donde tenemos instalado el JavaFX y no copiara todos los paquetes



- Asemos los mismos pasos en el apartado de “Run” solo que ahora le damos a al signo más en el apartado de Modulepatch

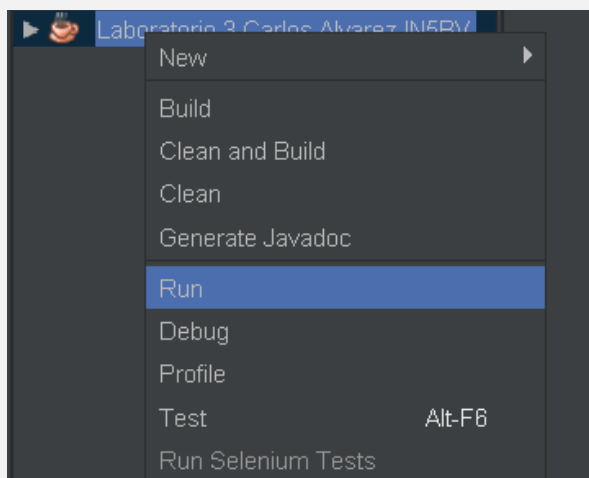


- Después nos dirigimos al apartado de “Run” del lado derecho y copiamos lo siguiente donde dice VM Options “--add-modules javafx.controls,javafx.fxml”

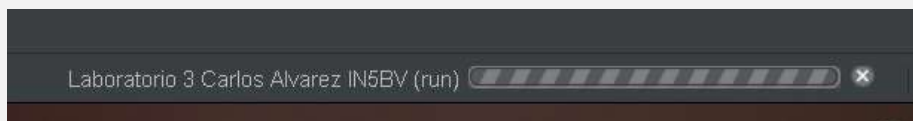


- Y ya lo tenemos configurado

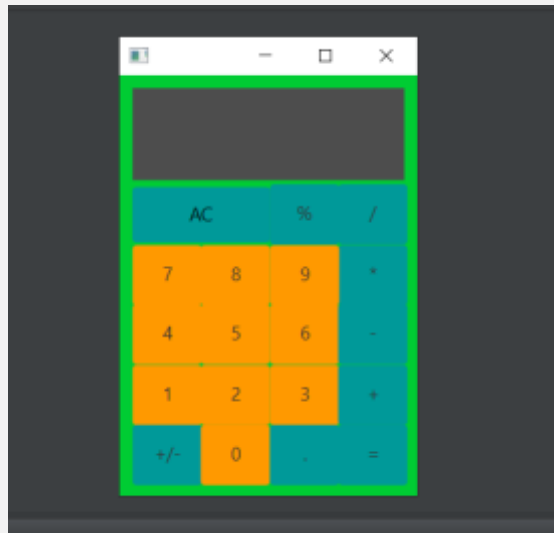
6. Al tenerlo abierto le damos clic derecho al proyecto y le damos donde dice “Run”



7. El proyecto se empezará a copilar y a ejecutarse (En este paso la compilación y la ejecución dependerá de los componentes que tengamos)



8. Se termina de compilar y se le aparecerá la calculadora y podemos empezar a hacer operaciones como los siguientes ejemplos



- Podemos hacer una suma sucesiva, resta, multiplicación, división

$$45 + 52 =$$

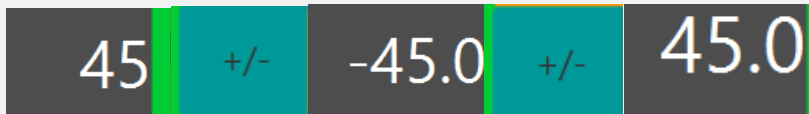
$$97.0 - 52 =$$

$$45.0 * 5 =$$



A digital calculator interface with a dark grey background. It shows the calculation  $225.0 \div 25 = 9.0$ . The numbers are in white, and the operators  $\div$  and  $=$  are in white on teal buttons. Green vertical bars are on the left and right edges.

- Podemos hacer que un numero se haga positivo o negativo



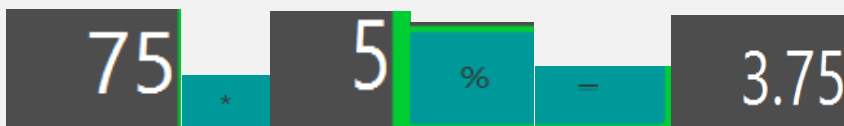
A digital calculator interface showing the sign change function. It starts with '45', then a teal button with '+/-' is pressed, resulting in '-45.0'. Pressing the '+/-' button again results in '45.0'. Green vertical bars are on the left and right edges.

- Podemos borrar todas las operaciones anteriores y empezar de nuevo



A digital calculator interface showing the 'All Clear' (AC) function. The display shows '45' and a teal button labeled 'AC' is highlighted. Green vertical bars are on the left and right edges.

- Y podemos calcular el promedio de un numero

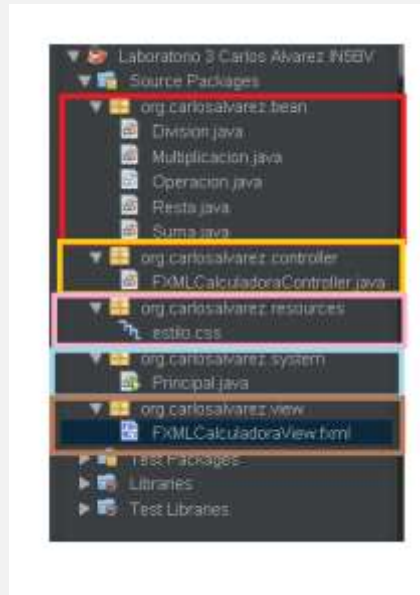


A digital calculator interface showing the calculation of the average. It starts with '75', then a teal button with '\*' is pressed, followed by '5', then a teal button with '%', and finally a teal button with '='. The result '3.75' is displayed. Green vertical bars are on the left and right edges.

## DESCRIPCIÓN DE LOS ALGORITMOS:

Empezaremos explicando lo más básico a los mas avanzado

Lo que explicaremos ahora son que función tiene cada paquete que vemos a continuación y que contiene adentro y como funciona



### 1. Paquete .bean:

El paquete que está encerrado con rojo, dentro del el podemos encontrar todas las operaciones que se pueden hacer dentro de la calculadora como vemos a simple vista están las operaciones básicas y hay una mas que se llama operaciones, esto es porque todas las operaciones tienen la misma “plantilla” que son dos variables y la operación y por eso aplicamos la herencia para hacer más rápido las operaciones Como vemos a continuación en la clase “Padre” Se hace lo siguiente:



```
// clase padre
public abstract class Operacion {

    //atributos
    private float numero1;
    private float numero2;
    private float resultado;
    private char operador;

    //constructores
    public Operacion() {
    }
}
```

Lo ponemos clase abstracta e inicializamos variable de tipo float ya que float hace los cálculos con menor número de decimales y además su operador de tipo char para saber que operación va hacer, y le ponemos su constructor vacío, a continuación, le ponemos los Getters y Setter:

```
public float getNumero1() {
    return numero1;
}

public void setNumero1(float numero1) {
    this.numero1 = numero1;
}

public float getNumero2() {
    return numero2;
}

public void setNumero2(float numero2) {
    this.numero2 = numero2;
}

public float getResultado() {
    return resultado;
}

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

public char getOperador() {
    return operador;
}

public void setOperador(char operador) {
    this.operador = operador;
}
}
```

En la siguiente imagen llamamos al método abstracto operar y al de String que es toString y adentro hacemos la ordenamos lo que nos vayan a pasar las clases hijas que son el numero1, numero2 y el operador y al final hacemos el cálculo de la operación

```

public abstract float operar(float numero1, float numero2);

@Override
public String toString() {
    return "**: " + this.getNumero1() + " ** + " + this.getOperador() + " ** + " + this.getNumero2() + " ** + " + this.getResultado();
}

```

Aquí ya tenemos hecho todo nuestro “molde” a hora hay que darles forma a las demás clases hijas (Solo hare un ejemplo con la clase su ya que las demás clases solo se cambia el signo)

```

public class Suma extends Operacion {

    @Override
    public float operar(float numero1, float numero2) {
        this.setNumero1(numero1);
        this.setNumero2(numero2);
        this.setOperador('+');
        this.setResultado(this.getNumero1() + this.getNumero2());
        return this.getResultado();
    }
}

```

Como vemos aquí en la clase suma tenemos que extenderle la clase operación para obtener todos los componentes de la clase operación, ya habiendo hecho todo esto llamamos al método abstracto operar, y le indicamos que le vamos a pasar dos números, el “this” que está ahí sirve para llamar al set del numero en la clase operación y dentro de los paréntesis esta la variable local que instanciamos dentro del método abstracto operar, en el this.operador miramos lo mismo solo que ahora con un char que es el signo “+” después de tener ya los dos números hacemos la operación, dentro del this.resultado mandamos a llamar con un get, el numero que acaba de ser settiado y hacemos la suma de los dos números, y al final retornamos el get del resultado (Para resta,

multiplicación y división son lo mismo lo único que cambia es el signo el método setOperador y el signo al hacer la operación en el método setResultado)

## 2. El paquete .controller:

Es el paquete que esta encerrado en anaranjado, dentro del están todos, pero todos los controladores de los botones de la calculadora, en este paquete le damos vida a la calculadora En principio vemos que tenemos que exportar todas estas librerías

```
import org.carlosalvarez.bean.Division;
import org.carlosalvarez.bean.Multiplicacion;
import org.carlosalvarez.bean.Resta;
import org.carlosalvarez.bean.Suma;
import java.net.URL;
import java.util.ResourceBundle;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import org.carlosalvarez.bean.Operation;
```

Las primeras 4 importaciones y la última son del paquete .bean, las demás importaciones son del JavaFX

Después vemos que se inicializa la clase y le extendemos el iniciador, también aquí tenemos que instanciar algunas variables que nos van a servir mas adelante, los ponemos como atributos privados y les realizamos sus get y set a cada uno

```

public class FMMCalculadoraContexto implements ICalculadora {

    private float resultado = 0.0f;
    private boolean esValido = true;
    private float numero = 0.0f;
    private int operadorActual = 0;
    private String operadorAnterior;
    private boolean esOperando = false;

    public float getResultado() {
        return resultado;
    }

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

    public float getNumero() {
        return numero;
    }

    public void setNumero(float numero) {
        this.numero = numero;
    }

    public int getOperadorActual() {
        return operadorActual;
    }

    public void setOperadorActual(int operadorActual) {
        this.operadorActual = operadorActual;
    }

    public String getOperador() {
        return operadorAnterior;
    }

    public void setOperador(String operador) {
        this.operadorAnterior = operador;
    }
}

```

A continuación, le pusimos un id a cada botón para poder identificar el botón

```

@FXML
private Label lblPantalla;
@FXML
private Button btnBorrar;
@FXML
private Button btnBorrarToda;
@FXML
private Button btnCero;
@FXML
private Button btnCeroDot;
@FXML
private Button btnComa;
@FXML
private Button btnIgual;
@FXML
private Button btnMasMenos;
@FXML
private Button btnMas;
@FXML
private Button btnMenos;
@FXML
private Button btnMulti;
@FXML
private Button btnDiv;
@FXML
private Button btnPorCien;
@FXML
private Button btnPar;
@FXML
private Button btnImpar;
@FXML
private Button btnFrac;
@FXML
private Button btnDec;
@FXML
private Button btnPor;
@FXML
private Button btnDiv2;
@FXML
private Button btnDiv3;
@FXML
private Button btnDiv4;
@FXML
private Button btnDiv5;

```

Pusimos un constructor vacío

```
@Override
public void initialize(URL url, ResourceBundle rb) {
    // TODO
}
```

A continuación, creamos métodos que servirán para darle funcionalidad a los botones

```
public void actualizarPantalla(String digito) {
    if (ingresoNuevo) {
        lblPantalla.setText("");
    }
    lblPantalla.setText(lblPantalla.getText().concat(digito));
    ingresoNuevo = false;
}
```

El método que tenemos aquí nos ayudara a poner números en el label

Aquí llamamos al método actualizar pantalla, y le pasamos su correspondiente número según el nombre del botón (Esto lo hicimos del número 0 al 10)

```
@FXML
private void clicSiete(ActionEvent event) {
    actualizarPantalla("7");
}
```

Ahora en el botón de porcentaje no llamamos al método actualizar pantalla, sino que hicimos la operación dentro del botón

```
@FXML
private void clicPorcentaje(ActionEvent event) {
    //lblPantalla.setText(lblPantalla.getText().concat("%"));
    resultado = resultado / 100;
}
```

Aquí agarramos el numero que resultado de la multiplicación de dos números y lo dividimos entre 100, ya que la operación del porcentaje nos dice multiplicar los dos números y dividirlos entre 100 y nos da el porcentaje

Aquí se creó un método para borrar todas las operaciones que s hayan hecho

```
private void limpiar() {
    lblPantalla.setText("");
    resultado = 0.0f;
    punto = true;
    numero = 0.0f;
    contadorOperadores = 0;
    operadorAnterior = "";
    contadorOperadores = 0;
    ingresoNuevo = false;
}
```

En esta parte inicializamos todas las variables a 0

Y ya en el clic borrar

```
@FXML
private void clicBorrar(ActionEvent event) {
    limpiar();
}
```

Solo llamamos al método que se creo arriba para darle funcionalidad al botón

Para poder convertir el numero en mas o menos solo lo multiplicamos por -1 y ya

```
@FXML
private void clicMasMenos(ActionEvent event) {
    resultado = [-1] * Float.parseFloat(lblPantalla.getText());
    lblPantalla.setText(String.valueOf(resultado));
}
```

Obtenemos lo que esta en pantalla lo multiplicamos por -1 y lo devolvemos en pantalla ya multiplicado por -1

Para el punto es algo complejo, ya que se deben evaluar varias opciones, si ya hay un punto existente, que ya no se agregue otro si, si no hay ningún numero y presionamos el punto que aparezca.0, y si es la primera vez que presionamos punto que, se agregue

```
@FXML
private void clicPunto(ActionEvent event) {
    if (ingresoNuevo) {
        lblPantalla.setText("");
    }

    String cadena = lblPantalla.getText();

    if (cadena.length() <= 0) {
        lblPantalla.setText("0.");
        ingresoNuevo = false;
    }

    if (lblPantalla.getText().contains(".")) {
        else {
            lblPantalla.setText(lblPantalla.getText() + ".");
            punto = false;
        }
    }
}
```

Ahora viene la parte que para mí es más importante porque aquí se le da la peculiaridad de la calculadora, que es hacer cálculos. A continuación miremos el código.

```
//operaciones
public void calcular(String operador) {

    if (!this.pantalla.getText().equals("")) {

        String strPantalla = this.pantalla.getText();

        if (ingresoNuevo == false) {

            this.numero = Float.parseFloat(strPantalla);

            contadorOperaciones++;

            if (contadorOperaciones == 1) {
                this.resultado = numero;
            } else if (contadorOperaciones == 2) {
                Operacion calc = null;
                switch (this.operadorAnterior) {
                    case "+":
                        calc = new Suma();
                        this.resultado = calc.operar(this.resultado, this.numero);
                        break;
                    case "-":
                        calc = new Resta();
                        this.resultado = calc.operar(this.resultado, this.numero);
                        break;
                    case "*":
                        calc = new Multiplicacion();
                        this.resultado = calc.operar(this.resultado, this.numero);
                        break;
                    case "/":
                        calc = new Division();
                        this.resultado = calc.operar(this.resultado, this.numero);
                        break;
                }

                this.pantalla.setText(String.valueOf(this.resultado));
            } // fin del else if

            // finaliza if ingreso nuevo

            this.operadorAnterior = operador;
        } // fin si verificado de espacio vacio
        ingresoNuevo = true;
    }
}
```

En el primer if miramos si la pantalla no está vacía, obtenemos el texto que está adentro de la pantalla, el segundo nos dice que si es un ingreso nuevo hacemos lo siguiente: el número lo convertimos en



float por que con eso haremos los cálculos y ponemos que la variable contadorOperadores valla aumentando, entramos en el siguiente if si el contadorOperadores es igual 1, el resultado va hace igual a numero y si contadorOperadores es igual o mayor a 2 entramos al switch para poder hacer las operaciones correspondientes,

```
switch (this.operadorAnterior) {  
    case "+":  
        calc = new Suma();  
        this.resultado = calc.operar(this.resultado, this.numero);  
        break;  
    case "-":  
        calc = new Resta();  
        this.resultado = calc.operar(this.resultado, this.numero);  
        break;  
    case "*":  
        calc = new Multiplicacion();  
        this.resultado = calc.operar(this.resultado, this.numero);  
        break;  
    case "/":  
        calc = new Division();  
        this.resultado = calc.operar(this.resultado, this.numero);  
        break;  
}
```

En el switch evaluamos el operadorAnterior, en el primer caso si presionamos "+", creamos un objeto de clase suma del paquete .bean y llamamos al .calc para calcular las operaciones y le pasamos los dos número para que se haga la operación correspondiente (Así con todas las operaciones siguientes, solo nos ubicamos en el caso correspondiente y creamos un objeto de cada operación del paquete .bean)

```

        this.resultado.setText(String.valueOf(this.resultado));
    } // Fin del else if

    // finaliza if ingreso nuevo

    this.operadorAnterior = operador;
} // Fin if verificador de espacio vacio
ingresoNuevo = true;
}

```

La primera llave cerramos el switch, la segunda cerramos el ifelse pero le indicamos que antes que se cierre mostremos la variable resultado en el label , se cierra el if del ingreso nuevo, no hacemos nada, el siguiente cerramos el if que verifica si hay un espacio en blanco, ahí solo le decimos que el operador anterior sea igual a operador, y al final cerramos el método y solo le decimos que ingreso nuevo sea igual a verdadero

Ahora viene la parte donde seleccionamos la operación que deseamos hacer

```

@FXML
private void clicMas(ActionEvent event) {
    calcular ("+" );
}

```

Cuando clicamos el botón mas se llamara al caso “+” del switch y se realizara el cálculo (Lo mismo hacemos con el botón menos, multiplicación y división, solo le cambiamos el signo según sea el caso)

Y al final solo mostramos lo que tenemos con el botón igual

```
@FXML
private void clicIgual(ActionEvent event) {
    calcular(String.valueOf(this.operadorAnterior));

    lblPantalla.setText(String.valueOf(this.resultado));
}
```

Llamamos al método calcular y lo convertimos en un String, después solo mostramos el resultado convertido en un String

### 3. Paquete .system:

En este paquete se encuentra la clase principal con el método main, aquí se encuentra el ejecutador del programa

Lo que hicimos de primero fue importar estas librerías

```
import javafx.application.Application;
import javafx.stage.Stage;
import javafx.scene.Parent;
import javafx.fxml.FXMLLoader;
import javafx.scene.Scene;
```

Estas nos sirven para poder ejecutar el JavaFX sin ningún error, estos sirven como un escenario donde pasan escenas y todo sigue un orden

Ahora vemos como esta construido la clase principal

```

public class Principal extends Application{

    public static void main(String[] args) {
        launch(args);
    }

    @Override

    public void start(Stage stage) throws Exception {
        Parent root = FXMLLoader.load(getClass().getResource("../view/FXMLCalculadoraView.fxml"));
        Scene scene = new Scene(root);
        stage.setScene(scene);
        stage.show();
    }
}

```

De primero miramos que tiene que estar extendida a la una clase "Application"

En el método main le debemos de poner la siguiente línea de código "launch(args);" por que esto nos ayuda a que el launcher de la aplicación se ejecute y nos pueda mostrar la interfaz grafica en la pantalla

A continuación escribimos el @Override para sobrescribir el método start, En la primera línea le pasamos donde esta ubicada nuestra interfaz gráfica, en nuestro caso esta ubicada en el paquete view, el de cual hablaremos mas adelante, y escribimos el nombre del archivo donde se encuentra la interfaz grafica en mi caso seria "../view/FXMLCalculadoraView.fxml"

En la segunda línea creamos una nueva escena, en la tercera creamos un nuevo escenario y le pasmos la escena que acabamos de crear

Y en la ultima line mostramos el escenario

Y listo...



Ahora veamos como diseñar la interfaz Gráfica:

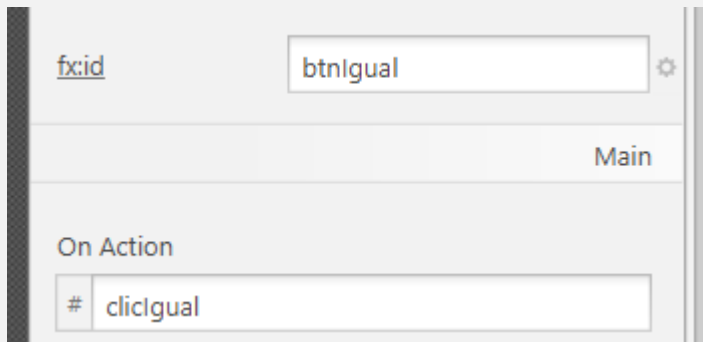
## DISEÑO:

1. Tener instalado SceneBuilder y abrir el archivo FXML que se genero al crear el proyecto JavaFX

Controller class

org.carlosalvarez.controller.FXMLCalculadoraControlle

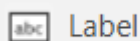
2. De aquí en adelante todo se hace automáticamente solo es de ir poniendo los objetos que nosotros queramos e ir poniéndole id y acciones




3. La forma de como se armo la calculadora fue la siguiente
  - Primero un anchorPane donde pondremos todos los objetos encima de el




- Colocamos un label en la parte superior del anchorPane




- Debajo del label colocamos un GridPane de 4\*5

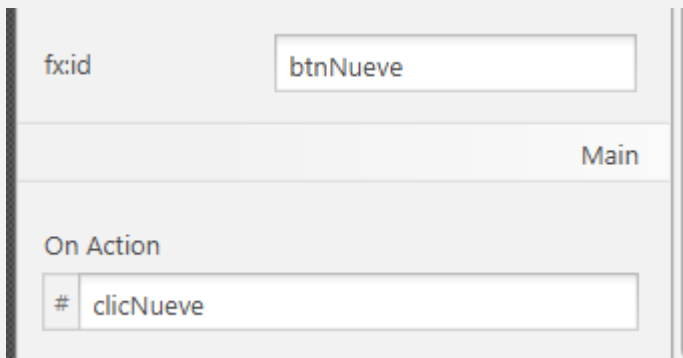
)  GridPane (4 x 5)

- Dentro de cada cuadrito del GridPane pondremos botones y les asignaremos números y al de las operaciones sus respectivos signos

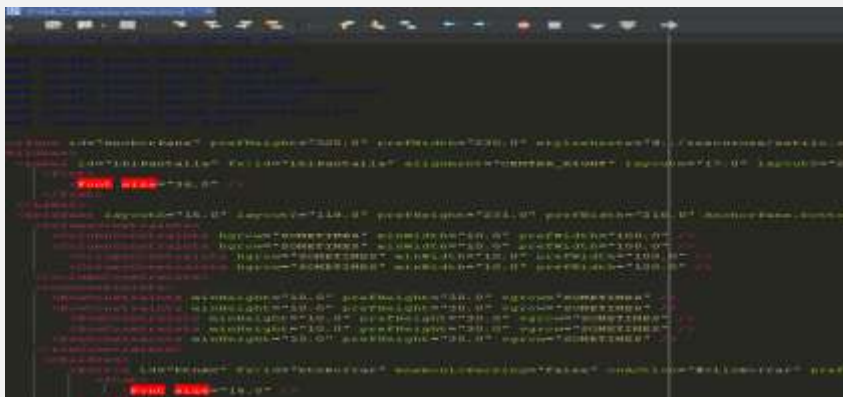
 Button (3, 0) /

 Button (0, 1) 7

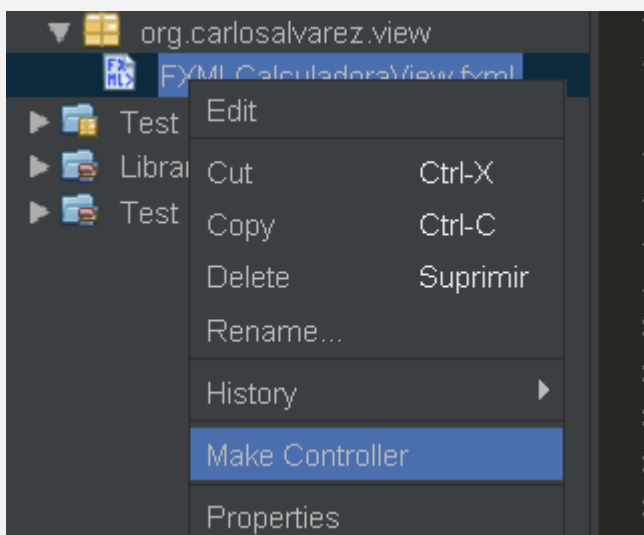
- Después de haber ordenado todo eso nos dirigimos a la parte de code del lado derecho le asignamos un id y una acción a cada objeto menos al anchorPane



4. Ya teniendo todo esto guardamos cambio y nos vamos al parte de código y vemos que todo esto se generó automáticamente



5. Dejamos ese código generado automática mente ahí ahora nos vamos a ese archivo clic derecho – make controller y se genera automáticamente todo el código para poder darle funcionalidad a los controles



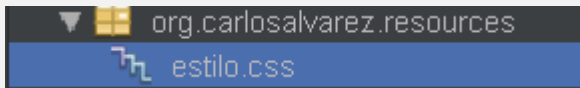


```

@FXML
private Label lblPantalla;
@FXML
private Button btnBorrar;
@FXML
private Button btnPresentar;
@FXML
private Button btnDividir;
@FXML
private Button btnIgual;
@FXML
private Button btnQuitar;
@FXML
private Button btnPasar;
@FXML
private Button btnMultiplicacion;
@FXML
private Button btnNuevo;
@FXML
private Button btnBorrar;
@FXML
private Button btnPasar;
@FXML
private Button btnPasar;
@FXML
private Button btnPasar;
@FXML
private Button btnPasar;
@FXML
private Button btnPasar;

```

6. En SceneBuilder podemos ponerle colores, pero si queremos poner mas colores mas opciones y mas cosas se debe hacer en un archivo css



7. En ese archivo para usarlo tenemos que ponerle el id del botón y con código poder decirle que color o que diseño ponerle

```

#btnIgual{
    -fx-background-color: #009999;
}

```

8. En SceneBuilder tenemos que irnos al apartado de la cosa que queremos modificar y nos vamos a la parte de Styleheets y buscamos el archivo css dentro de nuestro proyecto calculadora y se colocara lo que nosotros pusimos

Stylesheets

@	..\resources\estilo.css	+	▼
---	-------------------------	---	---



Y así con todos los objetos que queremos

“Cualquier tecnología suficientemente avanzada es equivalente a la magia” -Sir Arthur C. Clarke