



Unidad M3

Actividad 01

Variables constantes, cadenas y
operadores

Carlos Alberto Ramírez
Sánchez

No cuenta: 303044651

Programación de dispositivos móviles

Noviembre 2025



1. Descripción de la práctica

El objetivo de esta actividad es construir un programa de tipo calculadora utilizando interfaces de programación en Java. La calculadora debe:

- Realizar suma, resta, multiplicación y división.
- Usar al menos tres interfaces (ICalculadora, IProcesamiento, IDisplay, IControl).
- Validar división entre cero mostrando NaN o error.
- Mostrar el resultado en logcat o interfaz gráfica.

2. Estructura del proyecto

El proyecto está organizado según el patrón de arquitectura solicitado:



3. Interfaces implementadas

3.1. Interface IDisplay

```
1 package com.carlosramirez.micalculadora.calc;          1 1 ▲2 ✘5 ⌂ ⌂
2
3 /**
4  * Interface que define cómo se muestra el resultado o error.
5  */
6 no usages 1 implementation 1 related problem
7
8 public interface IDisplay {
9
10     no usages 1 implementation
11     String muestraResultado(float res);
12
13     no usages 2 related problems
14     String muestraError(CalculadoraError error);
15 }
16
17
```

3.2. Interface IPprocesamiento

```
r.java      iProcesamiento.java ✘  IDisplay.java    Procesamiento.java    ⌂ ⌂
1 package com.carlosramirez.micalculadora.calc;          ✓5 ⌂ ⌂
2
3 /**
4  * Interface que define las operaciones básicas de la calculadora.
5  */
6 no usages 1 implementation
7
8 public interface IPprocesamiento {
9
10     no usages 1 implementation
11     float suma(float x, float y);
12
13     no usages 1 implementation
14     float resta(float x, float y);
15
16     no usages 1 implementation
17     float multi(float x, float y);
18
19     no usages 1 implementation
20     float div(float x, float y);
21 }
```

3.3. Interface ICalculadora

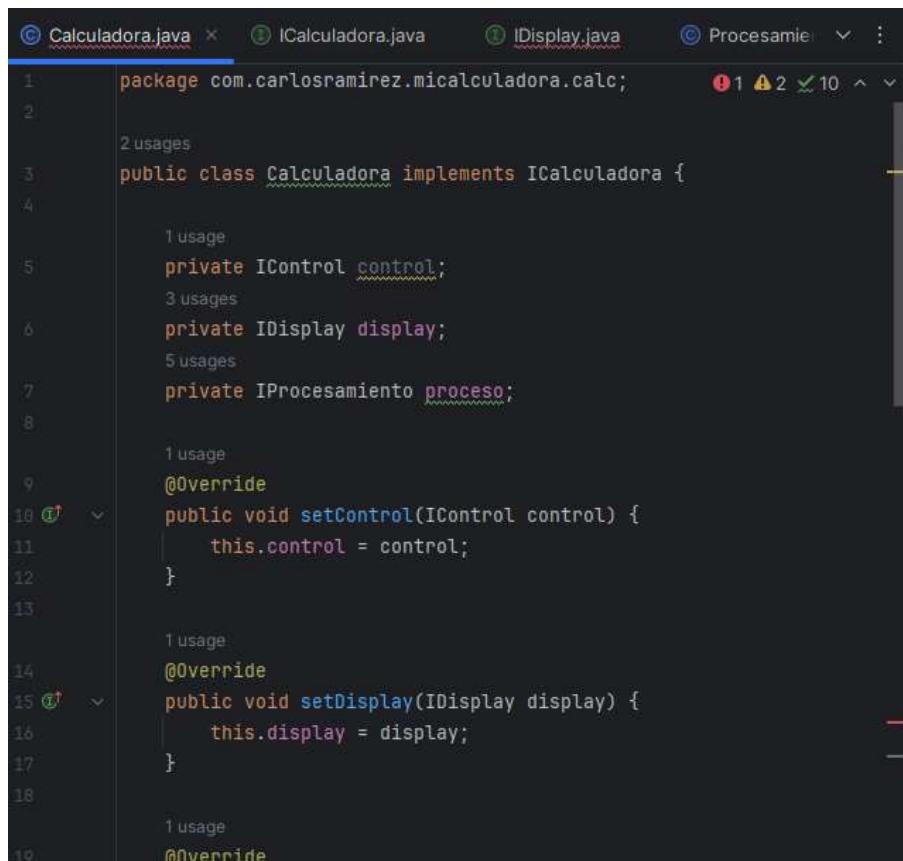
```
1 package com.carlosramirez.micalculadora.calc; ✓ 6
2
3 ① public interface ICalculadora {
4
5     1 usage 1 implementation
6     void setControl(IControl control);
7
8     1 usage 1 implementation
9     void setDisplay(IDisplay display);
10
11    1 usage 1 implementation
12    void setProcesamiento(IProcесamiento proceso);
13
14 }
```

4. Implementación de Procesamiento

```
dora.java ① ICalculadora.java ① IDisplay.java ② Procesamiento.java ✘ 7 ^ :
1 package com.carlosramirez.micalculadora.calc; ✓ 7 ^ :
2
3 public class Procesamiento implements IProcесamiento {
4
5     no usages
6     @Override
7     public float suma(float x, float y) {
8         return x + y;
9     }
10
11     no usages
12     @Override
13     public float resta(float x, float y) {
14         return x - y;
15     }
16
17     no usages
18     @Override
19     public float multi(float x, float y) {
20         return x * y;
21     }
22
23     no usages
24     @Override
25     public float div(float x, float y) {
```

```
no usages
28
29     @Override
30     public float div(float x, float y) {
31         // Evitar la división entre cero
32         if (y == 0) {
33             // Devolvemos NaN (Not a Number) para que el control/display
34             // puedan detectar el error y mostrarlo con CalculadoraError
35             return Float.NaN;
36         }
37         return x / y;
38     }
39 }
40
41 }
```

5. Implementación de Calculadora



The screenshot shows a Java code editor with several tabs at the top: Calculadora.java (selected), ICalculadora.java, IDisplay.java, and Procesamiento.java. The Calculadora.java tab shows the following code:

```
package com.carlosramirez.micalculadora.calc;

public class Calculadora implements ICalculadora {

    private IControl control;
    private IDisplay display;
    private IProcesamiento proceso;

    @Override
    public void setControl(IControl control) {
        this.control = control;
    }

    @Override
    public void setDisplay(IDisplay display) {
        this.display = display;
    }

    @Override
```

```
public class Calculadora implements ICalculadora {
    private IProcesamiento proceso;

    public void setProcesamiento(IProcesamiento proceso) {
        this.proceso = proceso;
    }

    @Override
    public String calcular(int x, int y, Operacion op) {
        float resultado = 0f;

        switch (op) {
            case SUM:
                resultado = proceso.suma(x, y);
                break;
            case RES:
                resultado = proceso.resta(x, y);
                break;
            case MUL:
                resultado = proceso.multi(x, y);
                break;
            case DIV:
                resultado = proceso.div(x, y);
                break;
        }
    }
}
```

```
35:         case MUL:
36:             resultado = proceso.multi(x, y);
37:             break;
38:         case DIV:
39:             resultado = proceso.div(x, y);
40:             break;
41:     }
42:
43:     // Manejo de errores: calculadoraError
44:     if (Float.isNaN(resultado)) {
45:         // Por ejemplo, división entre 0
46:         return display.muestraError(CalculadoraError.DIV_ZERO);
47:     }
48:
49:     return display.muestraResultado(resultado);
50: }
51}
52}
```

6. Diseño de la interfaz gráfica (XML)

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:padding="24dp">

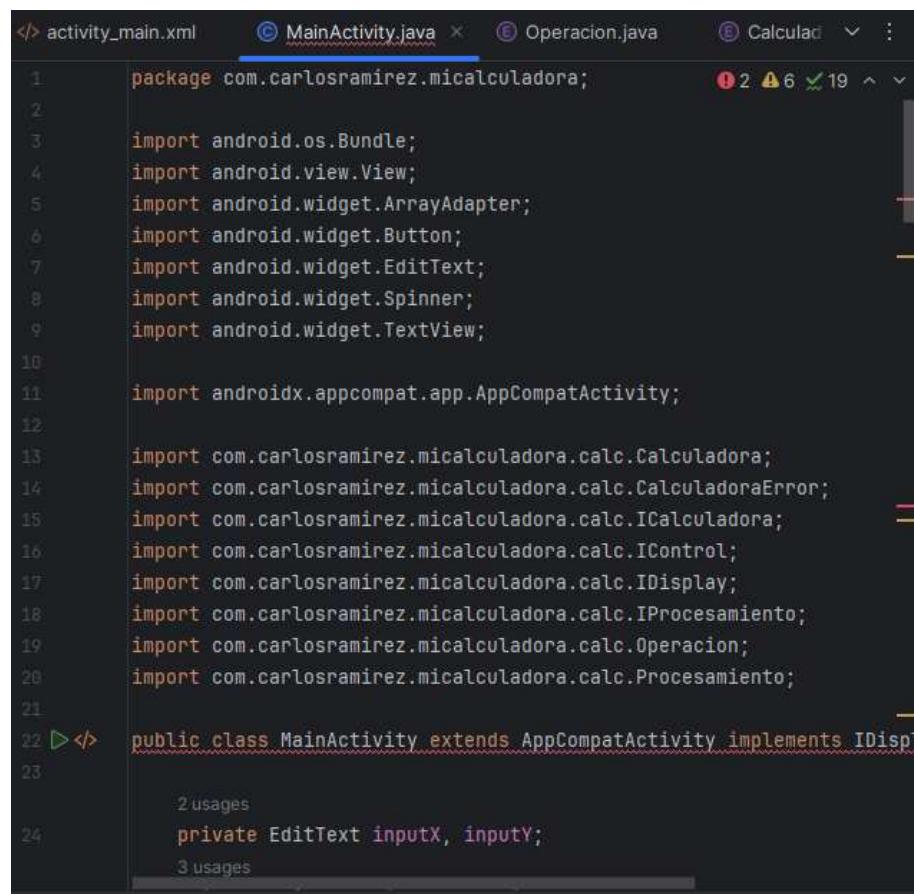
    <EditText
        android:id="@+id/inputX"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Valor X"
        android:inputType="number" />

    <EditText
        android:id="@+id/inputY"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Valor Y"
        android:inputType="number" />

    <Spinner
        android:id="@+id/spinnerOperacion"
        android:layout_width="match_parent"
        android:layout_height="wrap_content" />
```

```
26
27    <Button
28        android:id="@+id/btnCalcular"
29        android:layout_width="match_parent"
30        android:layout_height="wrap_content"
31        android:text="Calcular" />
32
33    <TextView
34        android:id="@+id/tvResultado"
35        android:layout_width="match_parent"
36        android:layout_height="wrap_content"
37        android:text="Resultado"
38        android:textSize="20sp"
39        android:paddingTop="16dp" />
40
41 </LinearLayout>
42
43
```

7. MainActivity (Controlador principal)



The screenshot shows the Android Studio code editor with the file `MainActivity.java` open. The code is part of an application named `micalculadora`. It imports various Android widgets and components from the `com.carlosramirez.micalculadora.calc` package. The class `MainActivity` extends `AppCompatActivity` and implements the `IDisplay` interface. It also contains private fields for `EditText` components `inputX` and `inputY`.

```
</> activity_main.xml   © MainActivity.java ×  ⏺ Operacion.java  ⏺ Calculad. ▾  ⏺
1 package com.carlosramirez.micalculadora;
2
3 import android.os.Bundle;
4 import android.view.View;
5 import android.widget.ArrayAdapter;
6 import android.widget.Button;
7 import android.widget.EditText;
8 import android.widget.Spinner;
9 import android.widget.TextView;
10
11 import androidx.appcompat.app.AppCompatActivity;
12
13 import com.carlosramirez.micalculadora.calc.Calculadora;
14 import com.carlosramirez.micalculadora.calc.CalculadoraError;
15 import com.carlosramirez.micalculadora.calc.IColorable;
16 import com.carlosramirez.micalculadora.calc.IDisplay;
17 import com.carlosramirez.micalculadora.calc.IProcesamiento;
18 import com.carlosramirez.micalculadora.calc.Operacion;
19 import com.carlosramirez.micalculadora.calc.Procesamiento;
20
21 ▷ </> public class MainActivity extends AppCompatActivity implements IDisp
22
23
24     2 usages
25     private EditText inputX, inputY;
26
27     3 usages
```

```
22     public class MainActivity extends AppCompatActivity { 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28     private ICalculadora calc; 2 usages 29     private IProcesamiento proceso; 30 31     @Override 32     protected void onCreate(Bundle savedInstanceState) { 33         super.onCreate(savedInstanceState); 34         setContentView(R.layout.activity_main); 35 36         // Referencias UI 37         inputX = findViewById(R.id.inputX); 38         inputY = findViewById(R.id.inputY); 39         spinnerOperacion = findViewById(R.id.spinnerOperacion); 40         tvResultado = findViewById(R.id.tvResultado); 41         Button btnCalcular = findViewById(R.id.btnCalcular); 42 43         // Modelo spinner 44         ArrayAdapter<String> adapter = new ArrayAdapter<>( 45             context, 46             android.R.layout.simple_spinner_item, 47             new String[]{"Suma", "Resta", "Multiplicación", "División"}); 48         adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item); 49         spinnerOperacion.setAdapter(adapter); 50 51 52     public class MainActivity extends AppCompatActivity implements IDisplay { 53         protected void onCreate(Bundle savedInstanceState) { 54 55             // Crear instancias 56             proceso = new Procesamiento(); 57             calc = new Calculadora(); 58             calc.setControl(this); 59             calc.setDisplay(this); 60             calc.setProcesamiento(proceso); 61 62             // Botón 63             btnCalcular.setOnClickListener(new View.OnClickListener() { 64                 @Override 65                 public void onClick(View v) { 66                     ejecutar(); 67                 } 68             }); 69 70             // ===== 71             // IDisplay 72             // ===== 73             1 usage 74             @Override 75             public String muestraResultado(float res) {
```

Pruebas

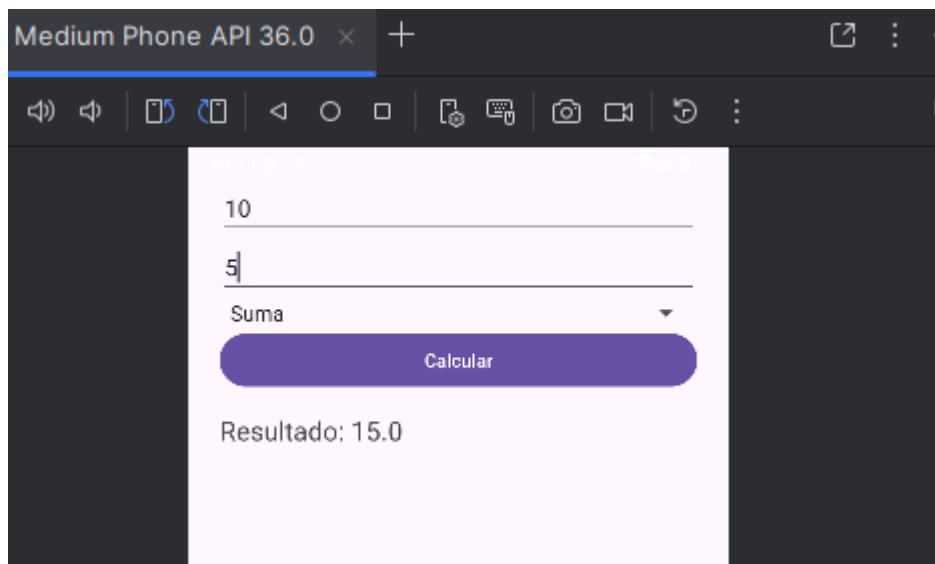
Prueba de suma

Entradas:

X = 10

Y = 5

Operación seleccionada: *Suma*



Prueba de resta

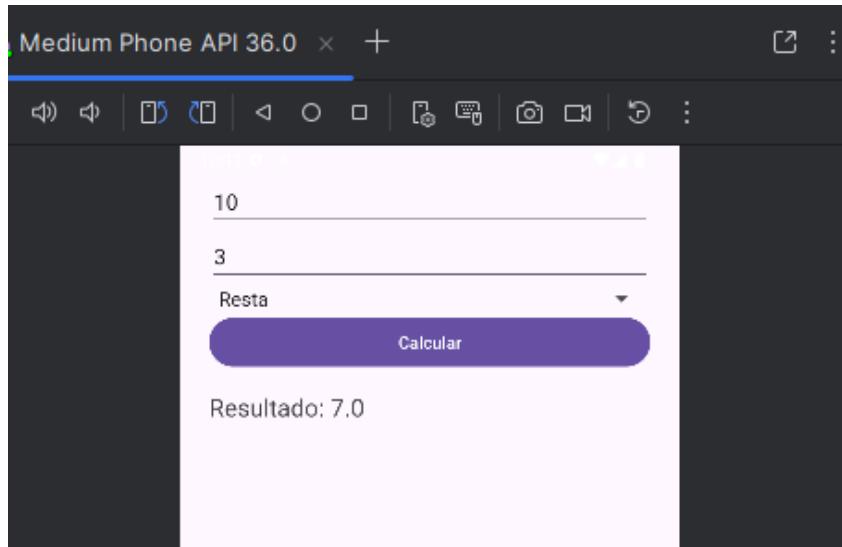
Entradas:

X = 10

Y = 3

Operación seleccionada: *Resta*

Resultado esperado: 7



Prueba de multiplicación

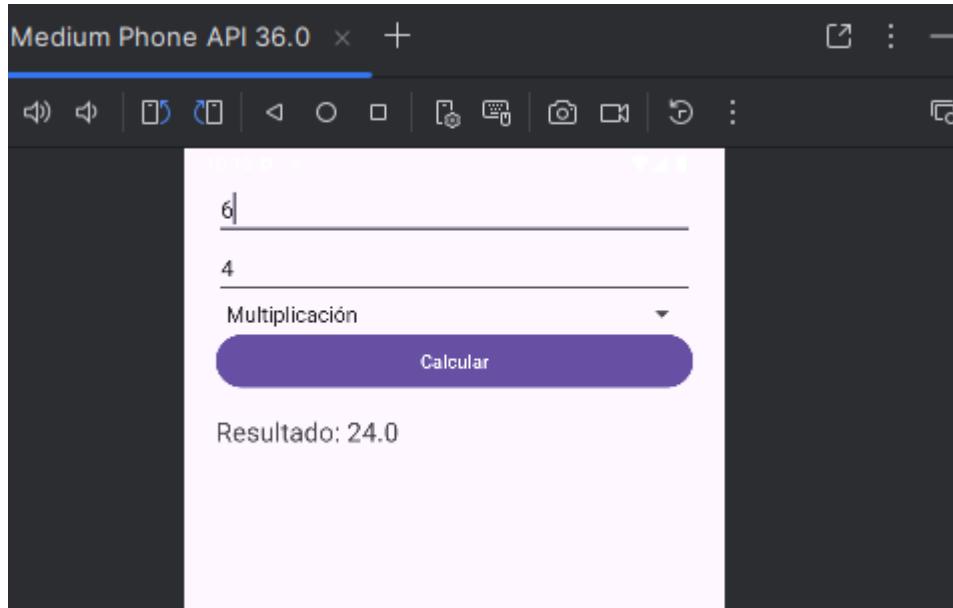
Entradas:

X = 6

Y = 4

Operación seleccionada: Multiplicación

Resultado esperado: 24



4. Prueba de división

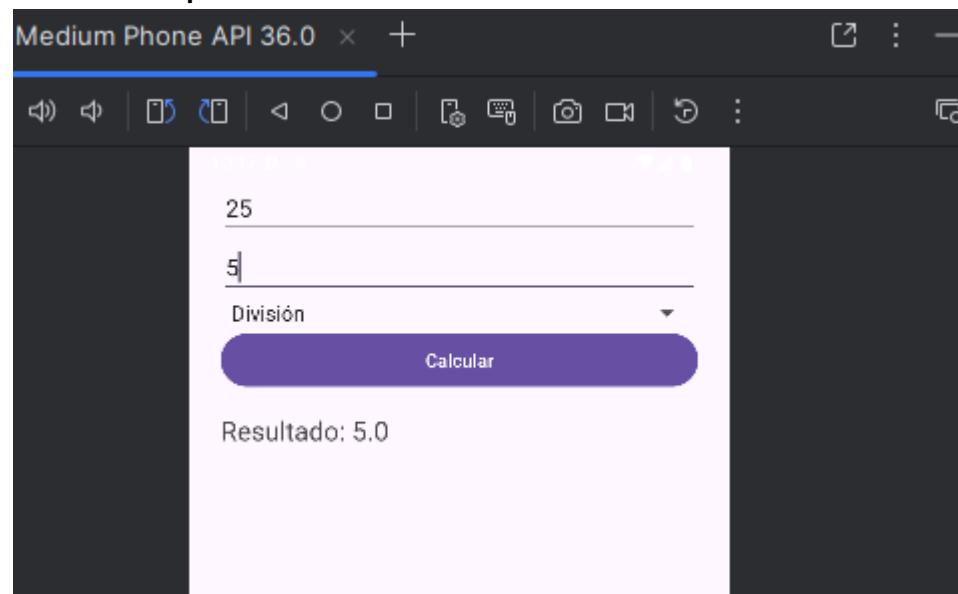
Entradas:

X = 20

Y = 5

Operación seleccionada: *División*

Resultado esperado: 4



5. Prueba de error: división entre cero

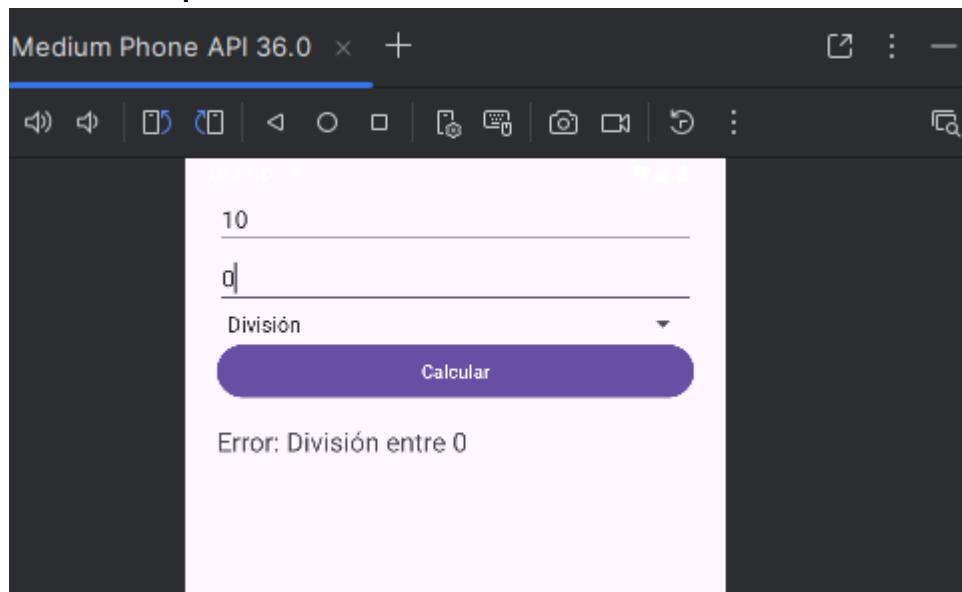
Entradas:

X = 10

Y = 0

Operación seleccionada: *División*

Resultado esperado: Error: División entre 0



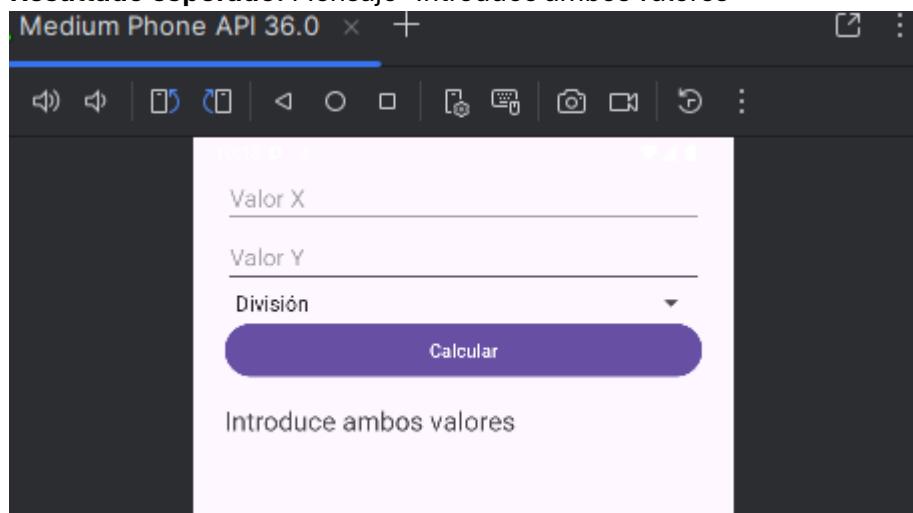
Prueba de campos vacíos

Entradas:

X vacío

Y vacío

Resultado esperado: Mensaje “Introduce ambos valores”



Conclusiones

El desarrollo de esta calculadora permitió poner en práctica los conceptos fundamentales de programación orientada a objetos vistos en la Unidad 3, especialmente el uso de **interfaces**, separación de responsabilidades y modularidad.

El programa se estructuró correctamente en capas:

- **IDisplay** para la salida de resultados y mensajes.
- **IControl** para recibir valores de entrada.
- **IProcesamiento** como capa lógica donde se realizan las operaciones matemáticas.
- **ICalculadora**, que integra todas las piezas y coordina el flujo.

Esta arquitectura ofrece varios beneficios:

1. **Escalabilidad:** permite agregar más operaciones sin modificar el código existente.
2. **Reusabilidad:** las clases de procesamiento pueden reutilizarse en otros proyectos.
3. **Mantenibilidad:** los cambios se realizan de forma localizada.
4. **Buen diseño:** se evita mezclar lógica, interfaz y control.

Además, se verificó que la calculadora cumple los requisitos del docente:

- Incluye suma, resta, multiplicación y división.
- Valida división entre cero (NaN).
- Utiliza interfaces correctamente implementadas.
- Muestra resultados y errores de manera clara.
- El flujo se observa correctamente en logcat, aunque también se adaptó una interfaz gráfica opcional para facilitar las pruebas.

Finalmente, las pruebas permiten concluir que el programa funciona correctamente, no presenta errores en tiempo de ejecución y respeta la arquitectura solicitada.

Referencias

Oracle. (s.f.). *Creating an interface*. Java Documentation.
<https://docs.oracle.com/javase/tutorial/java/IandI/createinterface.html>

Oracle. (s.f.). *Interfaces*. Java Documentation.
<https://docs.oracle.com/javase/tutorial/java/concepts/interface.html>

Oracle. (s.f.). *Questions and Exercises: Interfaces*. Java Documentation.
<https://docs.oracle.com/javase/tutorial/java/IandI/QandE/interfaces-questions.html>

Android Developers. (2024). *Android Developer Documentation*.
<https://developer.android.com/docs>

Bloch, J. (2018). *Effective Java* (3rd ed.). Addison-Wesley Professional.