

**UNIVERSIDAD ACADEMICA DE LA REGION NORTE DE GUERRERO UT**

**Docente; ING; Abel Jerónimo Vargas**

**Alumno:**

* **Roberto Chauteco Bello**
* **Documentación Act.7 Varios Activity**

**Chilapa de Álvarez Guerrero, OCTUBRE 2023**

**INDICE**

[Ilustración 1ARCHIVO MainActivity.Java 3](#_Toc150550911)

[Ilustración 2Activity\_main.xml 5](#_Toc150550912)

[Ilustración 3Archivo AndroidManifest.xml 7](#_Toc150550913)

[Ilustración 4APLICACIÓN CUADRADO 8](#_Toc150550914)

[Ilustración 5DISEÑO APLIACION CUADRADO 10](#_Toc150550915)

[Ilustración 6APLICACIÓN Octágono 12](#_Toc150550916)

[Ilustración 7DISEÑO APLIACION OCTAGONO 14](#_Toc150550917)

[Ilustración 8APLICACIÓN PIRAMIDE 16](#_Toc150550918)

[Ilustración 9DISEÑO APLIACION PIRAMIDE 19](#_Toc150550919)

[Ilustración 10FIGURA TRAPECIO 21](#_Toc150550920)

[Ilustración 11DISEÑO TRAPECIO 23](#_Toc150550921)

[Ilustración 12FIGURA Triangulo 25](#_Toc150550922)

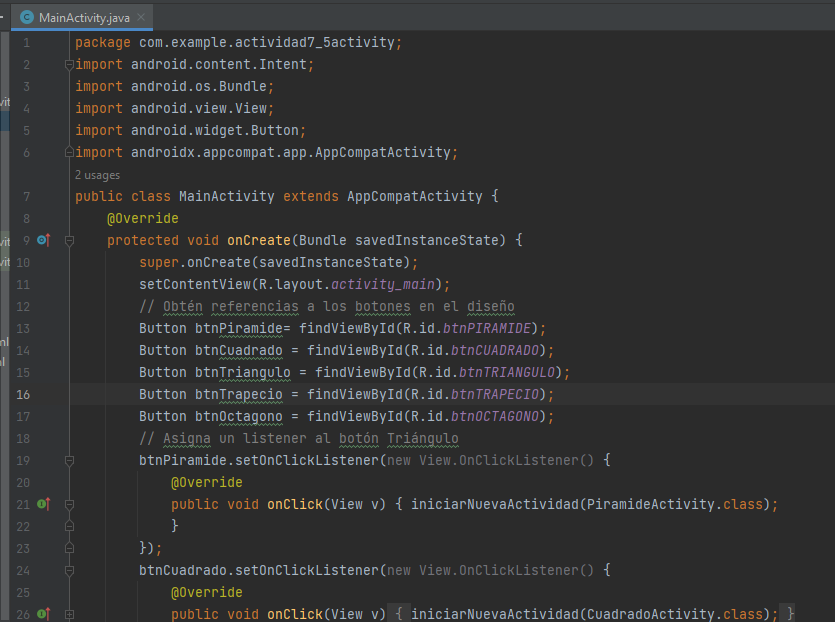
[Ilustración 13DISEÑO TRIANGULO 27](#_Toc150550923)

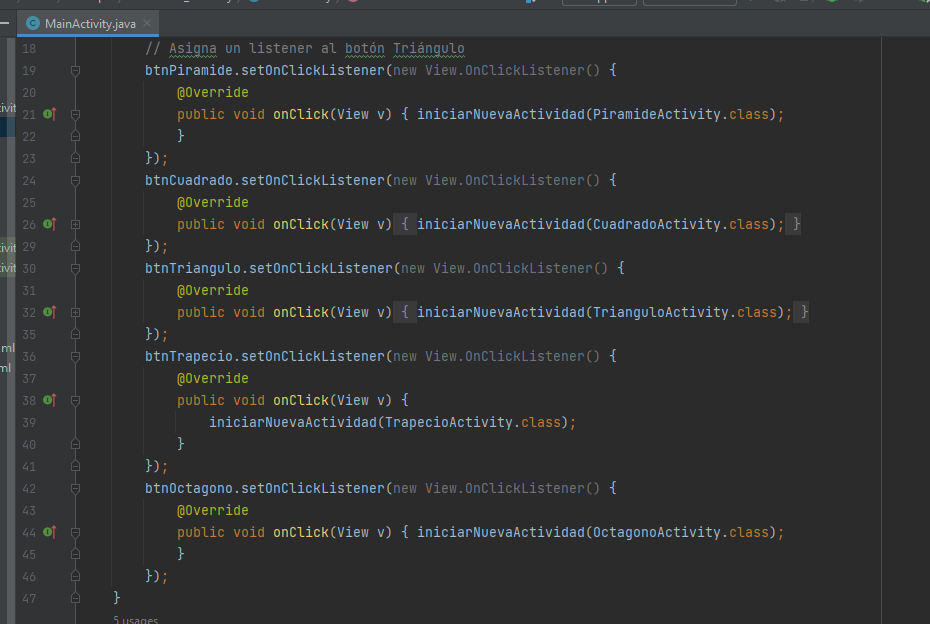
[Ilustración 14ARCHIVO DISEÑO MAIN.XML FINAL 29](#_Toc150550924)

# **ARCHIVO MainActivity.Java**

Ilustración 1ARCHIVO MainActivity.Java

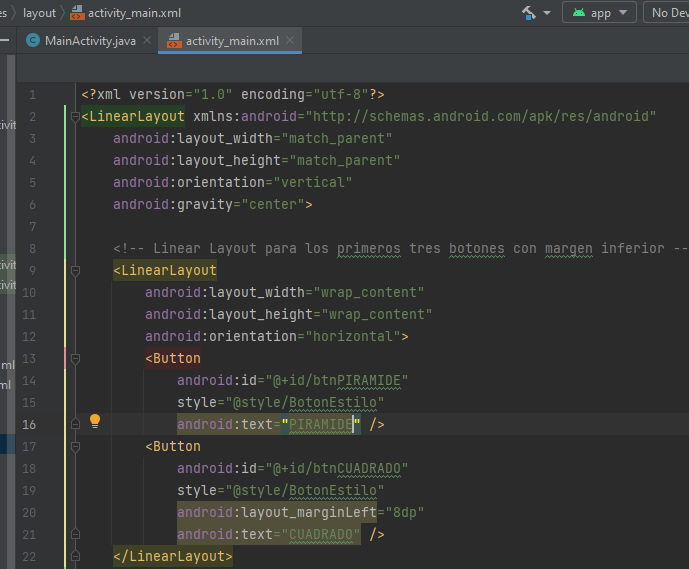
Dentro del creamos la Importación de las clases que nos ayudaran a que nuestra app funcione correctamente para la actividad, incluyendo las siguientes.

* Intent.
* Bundle
* View
* Button
*  AppCompatActivity.
* Posterior a las importaciones de las clases creamos la clase, MainActivity una creada la clase creamos el método onCreate, que se llama cuando la actividad se está iniciando. super. onCreate(savedInstanceState) es decir este método realiza cualquier inicialización necesaria por parte de la clase base, y setContentView (R. layout. activity\_main) establece el contenido de la actividad a partir del archivo de diseño XML llamado activity\_main.
* Dentro del método OnCreate Se obtienen cada uno de los botones en el diseño mediante sus identificadores (findViewById), es decir que con estos botones los definimos dentro del archivo de diseño activity\_main.xml.
* Le asignamos un listener al botón btnPiramide, para que cuando se hace haga clic en este botón, se llama al método iniciarNuevaActividad con la clase PiramideActivity.
* Le asignamos un listener al botón btnCuadrado, para que cuando se hace haga clic en este botón, se llama al método iniciarNuevaActividad con la clase CuadradoActivity.
* Le asignamos un listener al botón btnOctagono, para que cuando se hace haga clic en este botón, se llama al método iniciarNuevaActividad con la clase Octágono Activity.
* Le asignamos un listener al botón btnTrapecio, para que cuando se hace haga clic en este botón, se llama al método iniciarNuevaActividad con la clase TrapecioActivity
* Le asignamos un listener al botón btnTriangulo, para que cuando se hace haga clic en este botón, se llama al método iniciarNuevaActividad con la clase TrianguloActivity.

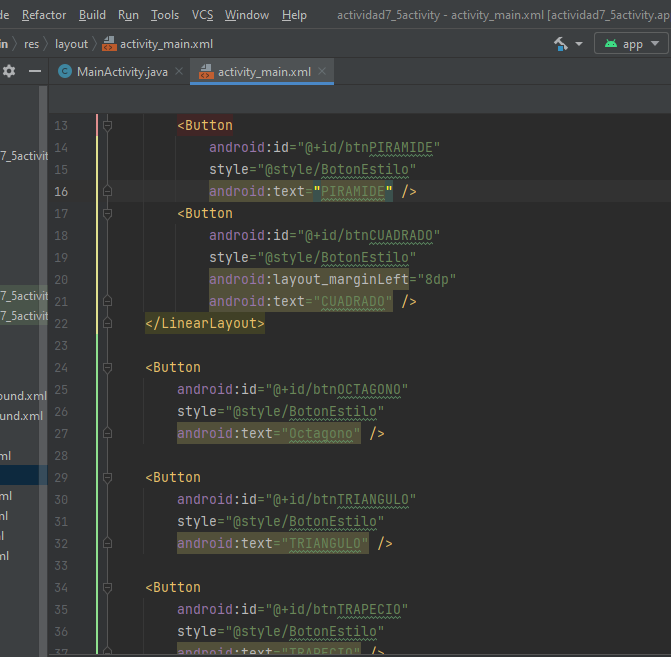


# **Activity\_main.xml**

Ilustración 2Activity\_main.xml

Dentro del archivo tenemos creamos la estructura principal de cada uno de los botones en que se visualizaran al inicio.

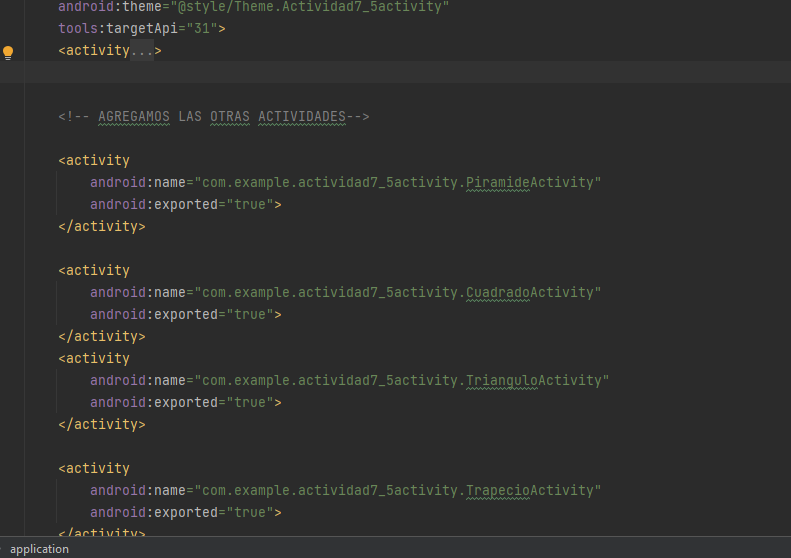
* Creamos un contenedor principal, un LinearLayout, que organiza sus elementos secuencialmente en una dirección (vertical en este caso), dentro del atributo gravity se establece en "center" para centrar los elementos horizontalmente que son principalmente los estilos principales.
* Creamos otro LinearLayout anidado que se utilizará para los primeros tres botones, en este LinearLayout tiene una orientación horizontal y se utilizará para alinear los botones PIRAMIDE y CUADRADO.
* Agregamos un botón con ID btnPIRAMIDE. Se aplica un estilo definido (@style/BotonEstilo) y se establece el texto del botón como "PIRAMIDE".
* Agregamos un al botón similar al anterior, se agrega otro botón con ID btnCUADRADO. También se aplica el mismo estilo, y se agrega un margen a la izquierda de 8dp para separar los botones horizontalmente.
* Agregamos un botón con ID btnOCTAGONO y estilo @style/BotonEstilo. El texto del botón se establece como "Octágono".
* Agregamos un botón con ID btnTRIANGULO y estilo @style/BotonEstilo. El texto del botón se establece como "TRIANGULO".
* Agregamos un botón con ID btnTRAPECIO y estilo @style/BotonEstilo. El texto del botón se establece como "TRAPECIO".



# **Archivo AndroidManifest.xml**

Ilustración 3Archivo AndroidManifest.xml

Dentro de la carpeta Android Manifest AndroidManifest.xml se agregaron diferentes activitys es decir aquí es donde agregamos cada uno de los botones de las aplicaciones para poder accede a ellos, es decir funciona de esta manera, si solo mandamos atraer y la visualización de las aplicaciones mediante los botones en el archivo main.java de nada sirve por qué no, tendrás acceso a la aplicación y para esto es necesario agregarle el activity que funciona básicamente como una ancla parecido como el que se utiliza frecuentemente en el lenguaje de HTML solo que este es en otro lenguaje de programación y funciona de la misma manera, y finalmente te da acceso a la aplicación siguiente.

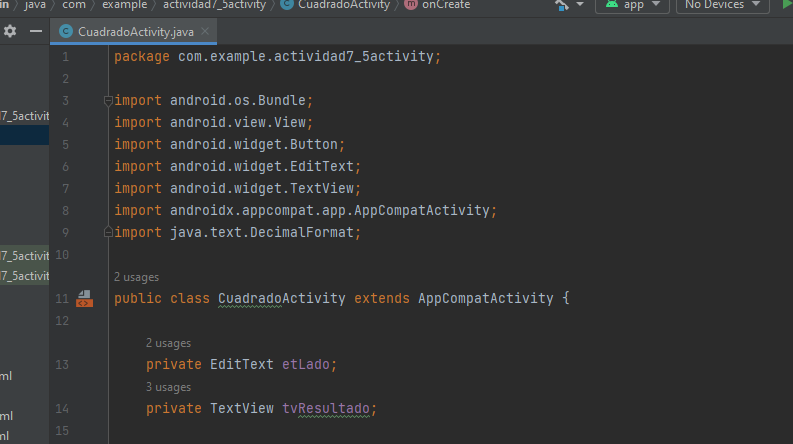


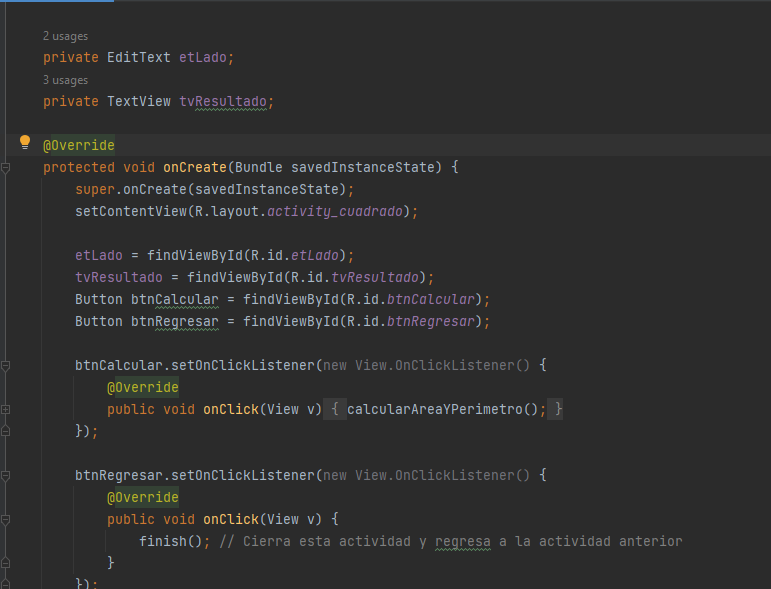
# **APLICACIÓN CUADRADO**

Ilustración 4APLICACIÓN CUADRADO

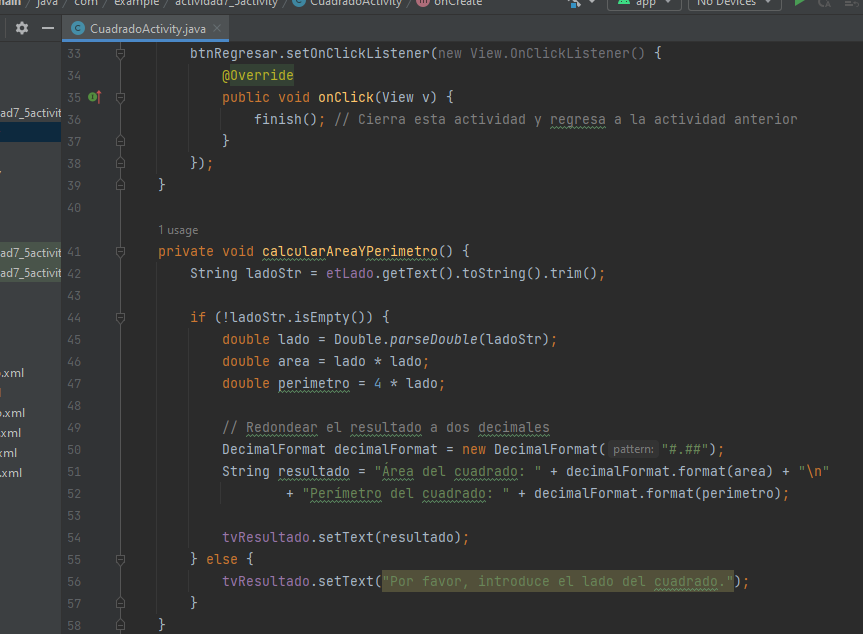
Dentro de este archivó CuadroActivity la figura tenemos en nombre la clase y principalmente tenemos las importaciones de las clases creadas para que nuestra aplicación funcione correctamente y estas son algunas importaciones que importamos al archivo CuadroActivity.

* import android.os.Bundle;
* import android.view.View;
* import android.widget.Button;
* import android.widget.EditText;
* import android.widget.TextView;
* import androidx.appcompat.app.AppCompatActivity;
* import java.text.DecimalFormat;



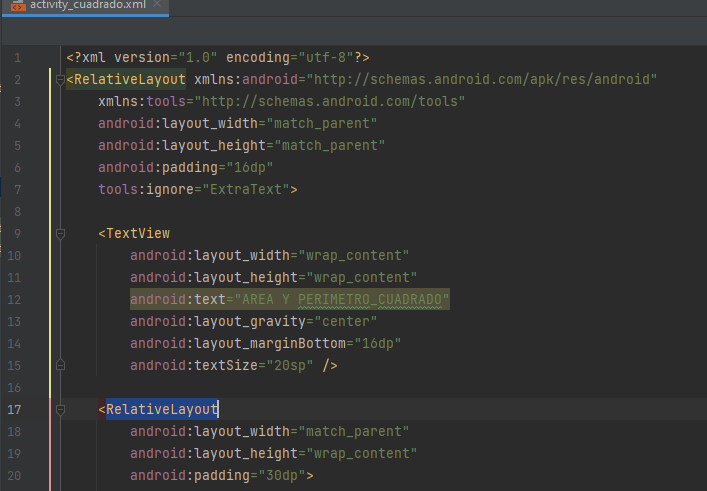


Después Declaración de la clase CuadradoActivity, que extiende AppCompatActivity,

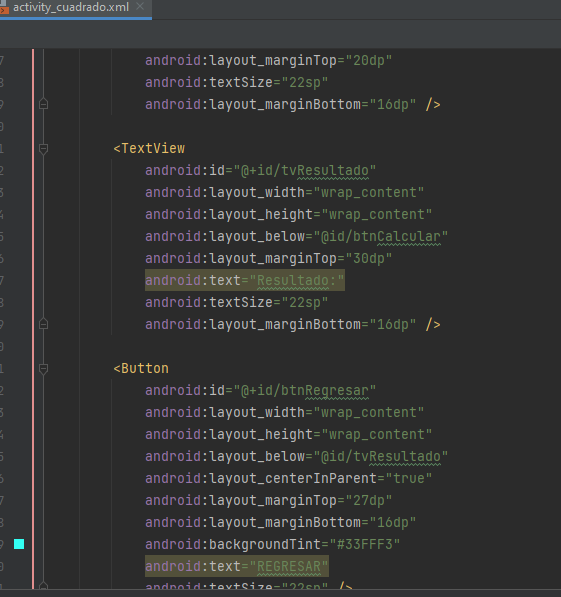
* private EditText etLado;
* private TextView tvResultado;
* Creamos el método OnCreate este método que se llama cuando la actividad se está iniciando. super.onCreate(savedInstanceState) realiza cualquier inicialización necesaria por parte de la clase base, y setContentView(R.layout.activity\_cuadrado) establece el contenido de la actividad a partir del archivo de diseño XML llamado activity\_cuadrado.
* Posteriormente obtenemos referencias de cada uno de los elementos de la interfaz de usuario mediante sus identificadores (findViewById), es decir estos elementos están definidos en el archivo de diseño activity\_cuadrado.xml.
* Asignamos un listener al botón btnCalcular. Cuando se hace clic en este botón, se llama al método calcularAreaYPerimetro.
* Se asigna un listener al botón btnRegresar. Cuando se hace clic en este botón, se llama al método finish (), que cierra esta actividad y regresa a la actividad anterior.
*  Por último, definimos el método calcularAreaYPerimetro, que realiza el cálculo del área y el perímetro del cuadrado. Se obtiene el valor del lado desde etLado (un EditText). Si el valor no está vacío, se realiza el cálculo y se muestra el resultado formateado en tvResultado (un TextView). Si el valor del lado está vacío, se muestra un mensaje indicando al usuario que debe ingresar el lado del cuadrado.

# **DISEÑO APLIACION CUADRADO**

Ilustración 5DISEÑO APLIACION CUADRADO

Dentro del archivo activity.cuadrado.xml creamos un contenedor principal, un RelativeLayout, que permite organizar los elementos de la interfaz de usuario de manera relativa.

* Se agrega un TextView que muestra el título "AREA Y PERIMETRO\_CUADRADO". Se configuran atributos como el tamaño del texto, el margen inferior para centrar el texto.
* Agregamos otro RelativeLayout anidado que se utilizará para organizar los elementos internos con un margen interno de 30dp.
* Agregamos un EditText con ID etLado para que el usuario ingrese el lado del cuadrado. Se establece un mensaje de pista (hint), un tipo de entrada numérico (numberDecimal), un tamaño de texto y un margen inferior.
* Se agrega un botón con ID btnCalcular que permite al usuario calcular el área y el perímetro. Se establece su posición debajo del EditText, el texto del botón, y se configuran tamaños y márgenes.
* Se agrega otro TextView con ID tvResultado para mostrar el resultado del cálculo. Se establece su posición debajo del botón de calcular, el texto, el tamaño del texto y los márgenes.
* Se agrega un botón con ID btnRegresar que permite al usuario regresar. Se establece su posición debajo del tvResultado, en el centro del diseño, el color de fondo (backgroundTint), el texto del botón y se configuran tamaños y márgenes.

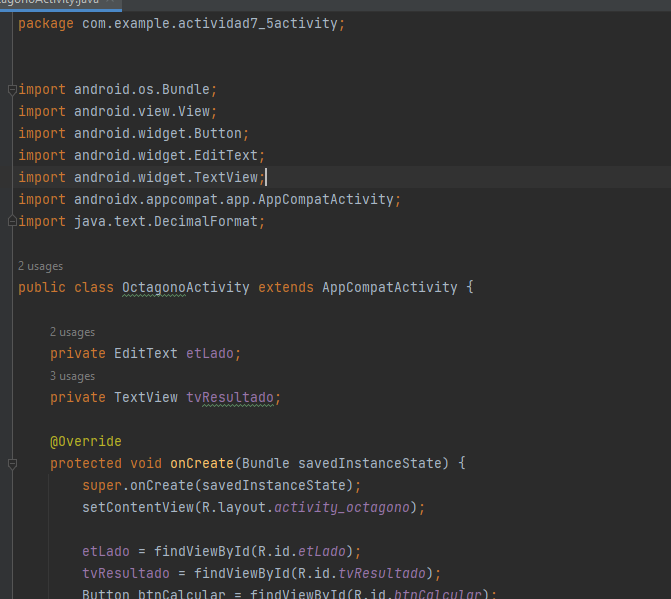


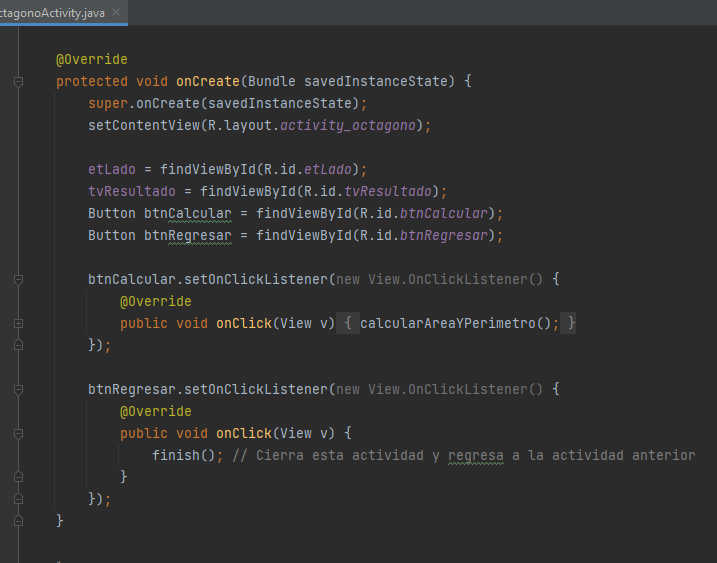
**APLICACIÓN Octágono**

Ilustración 6APLICACIÓN Octágono

Dentro de este archivó Octágono Activity la figura tenemos en nombre la clase y principalmente tenemos las importaciones de las clases creadas para que nuestra aplicación funcione correctamente y estas son algunas importaciones que importamos al archivo Octágono Activity.

* import android.os.Bundle;
* import android.view.View;
* import android.widget.Button;
* import android.widget.EditText;
* import android.widget.TextView;
* import androidx.appcompat.app.AppCompatActivity;
* import java.text.DecimalFormat;



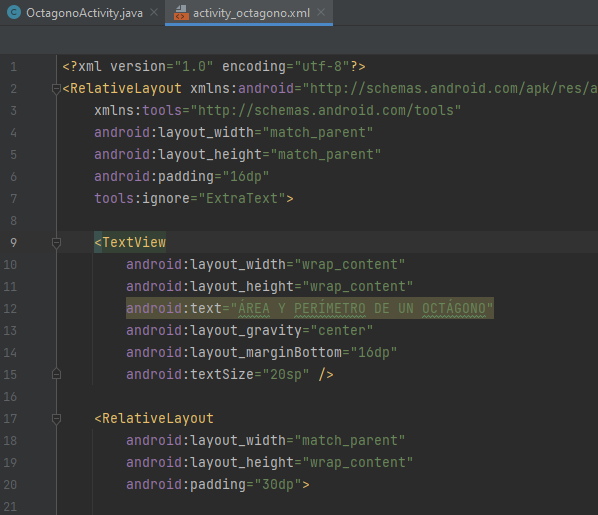


Después Declaración de la clase Octágono Activity, que extiende AppCompatActivity,

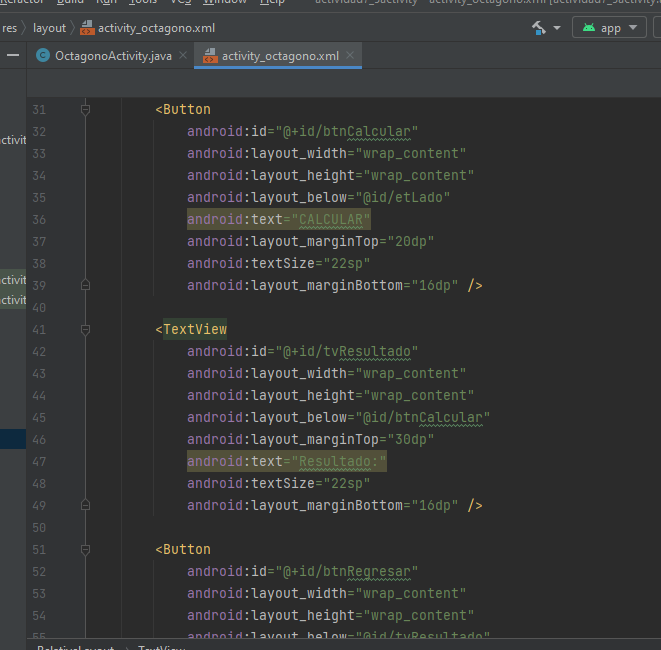
* private EditText etLado;
* private TextView tvResultado;
* Creamos el método OnCreate este método que se llama cuando la actividad se está iniciando. super.onCreate(savedInstanceState) realiza cualquier inicialización necesaria por parte de la clase base, y setContentView(R.layout.activity\_cuadrado) establece el contenido de la actividad a partir del archivo de diseño XML llamado activity\_ Octágono.
* Posteriormente obtenemos referencias de cada uno de los elementos de la interfaz de usuario mediante sus identificadores (findViewById), es decir estos elementos están definidos en el archivo de diseño activity\_ Octágono.xml.
* Asignamos un listener al botón btnCalcular, cuando se hace clic en este botón, se llama al método calcularAreaYPerimetro.
* Se asigna un listener al botón btnRegresar, cuando se hace clic en este botón, se llama al método finish (), que cierra esta actividad y regresa a la actividad anterior.
*  Por último, definimos el método calcularAreaYPerimetro, que realiza el cálculo del área y el perímetro del octágono. Se obtiene el valor del lado desde etLado (un EditText). Si el valor no está vacío, se realiza el cálculo y se muestra el resultado formateado en tvResultado (un TextView). Si el valor del lado está vacío, se muestra un mensaje indicando al usuario que debe ingresar el lado del Octágono.

# **DISEÑO APLIACION OCTAGONO**

Ilustración 7DISEÑO APLIACION OCTAGONO

Dentro del archivo activity. octágono.xml creamos un contenedor principal, un RelativeLayout, que permite organizar los elementos de la interfaz de usuario de manera relativa.

* Se agrega un TextView que muestra el título "AREA Y PERIMETRO\_OCTAGONO ". Se configuran atributos como el tamaño del texto, el margen inferior para centrar el texto.
* Agregamos otro RelativeLayout anidado que se utilizará para organizar los elementos internos con un margen interno de 30dp.
* Agregamos un EditText con ID etLado para que el usuario ingrese el lado del Octágono, Se establece un mensaje de pista (hint), un tipo de entrada numérico (numberDecimal), un tamaño de texto y un margen inferior.
* Se agrega un botón con ID btnCalcular que permite al usuario calcular el área y el perímetro. Se establece su posición debajo del EditText, el texto del botón, y se configuran tamaños y márgenes.
* Se agrega otro TextView con ID tvResultado para mostrar el resultado del cálculo. Se establece su posición debajo del botón de calcular, el texto, el tamaño del texto y los márgenes.
* Se agrega un botón con ID btnRegresar que permite al usuario regresar. Se establece su posición debajo del tvResultado, en el centro del diseño, el color de fondo (backgroundTint), el texto del botón y se configuran tamaños y márgenes.

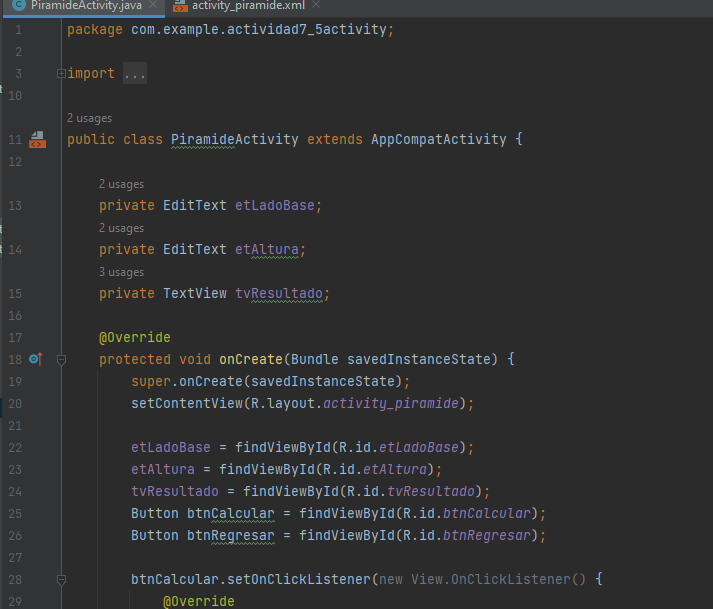


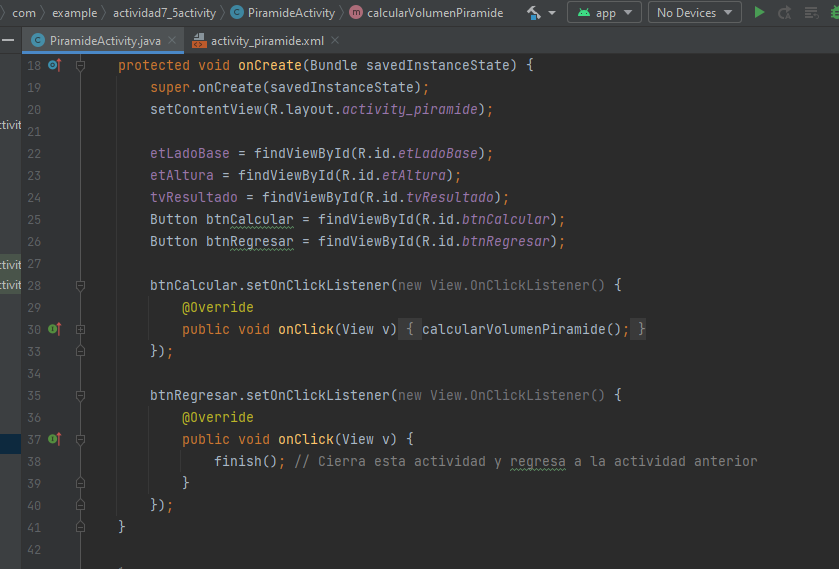
# **APLICACIÓN PIRAMIDE**

Ilustración 8APLICACIÓN PIRAMIDE

Dentro de este archivó PiramideActivity la figura tenemos en nombre la clase y principalmente tenemos las importaciones de las clases creadas para que nuestra aplicación funcione correctamente y estas son algunas importaciones que importamos al archivo PiramideActivity.

* import android.os.Bundle;
* import android.view.View;
* import android.widget.Button;
* import android.widget.EditText;
* import android.widget.TextView;
* import androidx.appcompat.app.AppCompatActivity;
* import java.text.DecimalFormat;





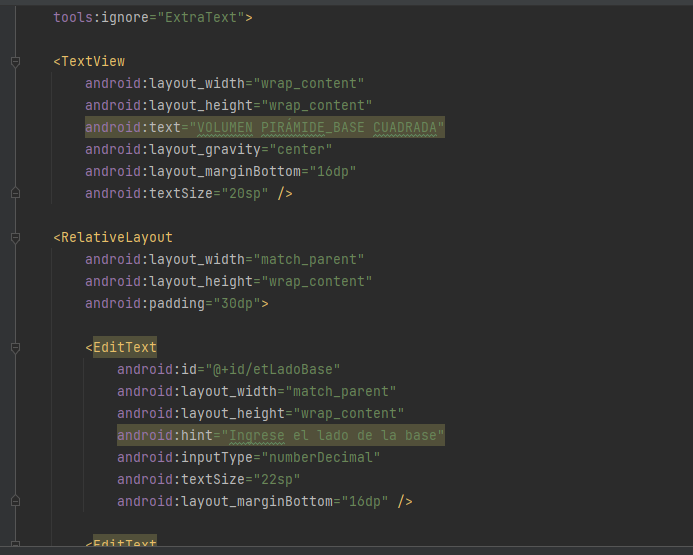
Después Declaración de la clase PiramideActivity, que extiende AppCompatActivity,

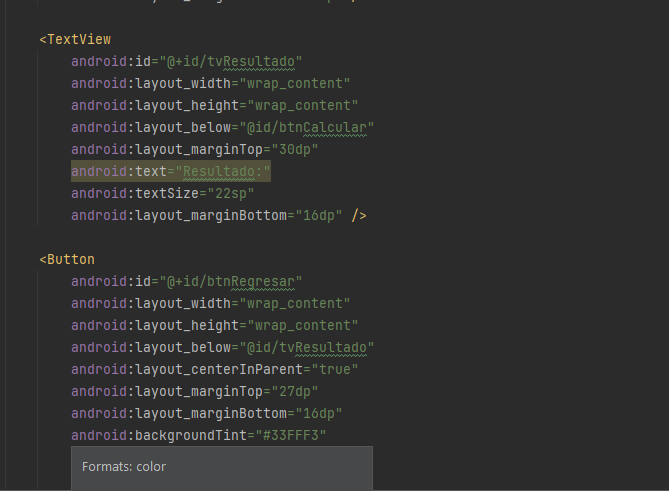
* private EditText etLadoBase;
* private EditText etAltura;
* private TextView tvResultado;
* Creamos el método OnCreate este método que se llama cuando la actividad se está iniciando. super.onCreate(savedInstanceState) realiza cualquier inicialización necesaria por parte de la clase base, y setContentView(R.layout.activity\_piramide) establece el contenido de la actividad a partir del archivo de diseño XML llamado activity\_ pirámide.
* Posteriormente obtenemos referencias de cada uno de los elementos de la interfaz de usuario mediante sus identificadores (findViewById), es decir estos elementos están definidos en el archivo de diseño activity\_ pirámide.xml.
* Asignamos un listener al botón btnCalcular, cuando se hace clic en este botón, se llama al método calcular EtLadobase.
* Asignamos un listener al botón btnCalcular, cuando se hace clic en este botón, se llama al método calcular EtAltura.
* Se asigna un listener al botón btnRegresar, cuando se hace clic en este botón, se llama al método finish (), que cierra esta actividad y regresa a la actividad anterior.
* Por último, definimos el método calcular Volumen de una pirámide, que realiza el cálculo del Volumen de la pirámide. Se obtiene el valor del lado desde etLado (un EditText). Si el valor no está vacío, se realiza el cálculo y se muestra el resultado formateado en tvResultado (un TextView). Si el valor del lado está vacío, se muestra un mensaje indicando al usuario que debe ingresar el lado y la altura de la pirámide.
* Y además se crea la función y operaciones para la creación del resultado del volumen de la pirámide formula volumen = (ladoBase \* ladoBase \* altura) / 3;

# **DISEÑO APLIACION PIRAMIDE**

Ilustración 9DISEÑO APLIACION PIRAMIDE

Dentro del archivo activity. piramide.xml creamos un contenedor principal, un RelativeLayout, que permite organizar los elementos de la interfaz de usuario de manera relativa



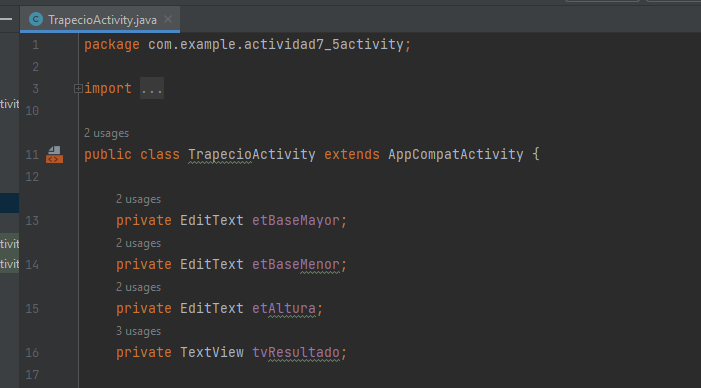
* Se agrega un TextView que muestra el título "Ingresar el lado de la base". Se configuran atributos como el tamaño del texto, el margen inferior para centrar el texto.
* Se agrega un TextView que muestra el título "Ingresar el lado de la altura". Se configuran atributos como el tamaño del texto, el margen inferior para centrar el texto.
* Agregamos otro RelativeLayout anidado que se utilizará para organizar los elementos internos con un margen interno de 30dp.
* Agregamos un EditText con ID etLado para que el usuario ingrese el lado del base, Se establece un mensaje de pista (hint), un tipo de entrada numérico (numberDecimal), un tamaño de texto y un margen inferior.
* Agregamos un EditText con ID etLado para que el usuario ingrese el lado de la altura, Se establece un mensaje de pista (hint), un tipo de entrada numérico (numberDecimal), un tamaño de texto y un margen inferior.
* Se agrega un botón con ID btnCalcular calcular volumen de la pirámide Se establece su posición debajo del EditText, el texto del botón, y se configuran tamaños y márgenes.
* Se agrega otro TextView con ID tvResultado para mostrar el resultado del cálculo. Se establece su posición debajo del botón de calcular, el texto, el tamaño del texto y los márgenes.
* Se agrega un botón con ID btnRegresar que permite al usuario regresar. Se establece su posición debajo del tvResultado, en el centro del diseño, el color de fondo (backgroundTint), el texto del botón y se configuran tamaños y márgenes.

# **TRAPECIO**

Ilustración 10FIGURA TRAPECIO

Dentro de este archivó TrapecioActivity la figura tenemos en nombre la clase y principalmente tenemos las importaciones de las clases creadas para que nuestra aplicación funcione correctamente y estas son algunas importaciones que importamos al archivo TrapecioActivity.

* import android.os.Bundle;
* import android.view.View;
* import android.widget.Button;
* import android.widget.EditText;
* import android.widget.TextView;
* import androidx.appcompat.app.AppCompatActivity;
* import java.text.DecimalFormat;



Declaración de la clase TrapecioActivity, que extiende AppCompatActivity.

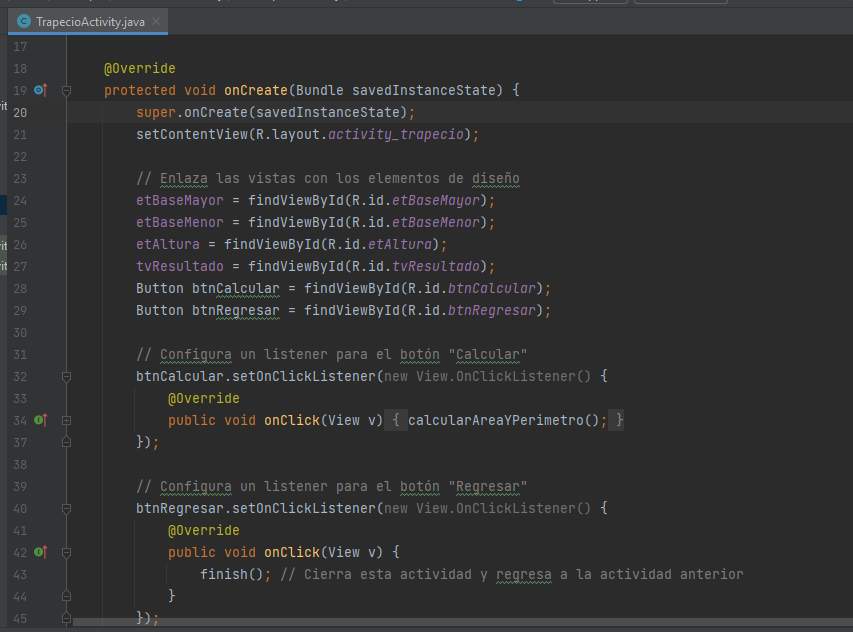
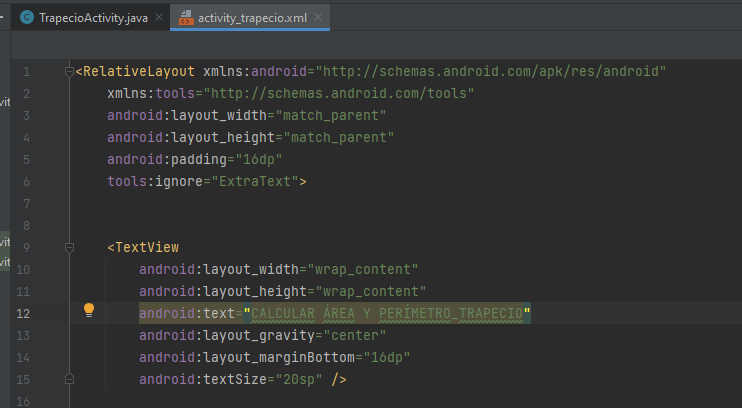
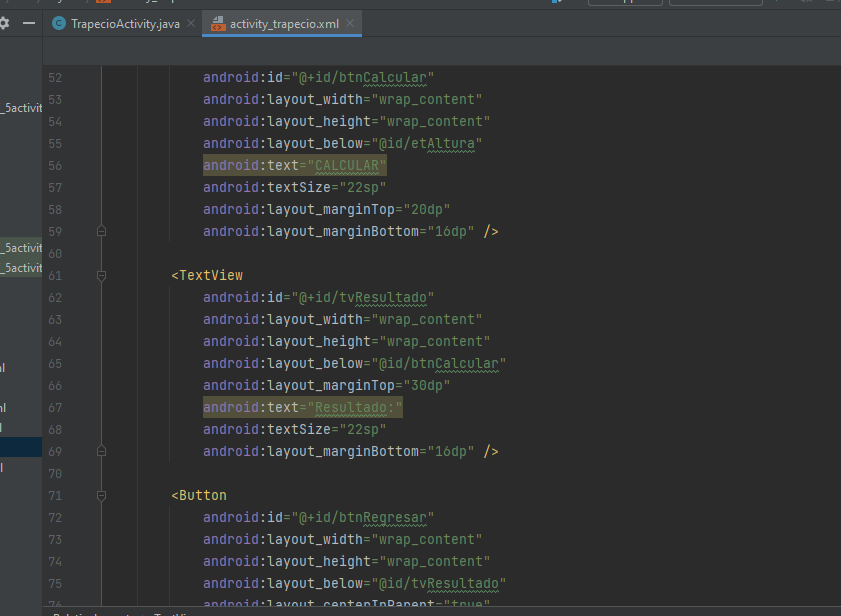
* Declaración de variables (etBaseMayor, etBaseMenor, etAltura y tvResultado) para referenciar los elementos de la interfaz de usuario.
* Iniciamos el método onCreate, que es llamado cuando la actividad se está iniciando. super. onCreate(savedInstanceState) realiza cualquier inicialización necesaria por parte de la clase base, y setContentView(R.layout.activity\_trapecio) establece el contenido de la actividad a partir del archivo de diseño XML llamado activity\_trapecio.
* Se obtienen referencias a los elementos de la interfaz de usuario mediante sus identificadores (findViewById), estos elementos probablemente estén definidos en el archivo de diseño activity\_trapecio.xml.
* Se asigna un listener al botón btnCalcular, cuando se hace clic en este botón, se llama al método calcularAreaYPerimetro.
* Se agrega un listener al botón btnRegresar, Cuando se hace clic en este botón, se llama al método finish(), que cierra esta actividad y regresa a la actividad anterior.
* Definimos el método calcularAreaYPerimetro, que realiza el cálculo del área y el perímetro del trapecio, y dentro de ellos se crean las operaciones de los valores de las bases y la altura desde etBaseMayor, etBaseMenor y etAltura (tres EditText), con la formula.
* area = (baseMayor + baseMenor) \* altura / 2;
* perimetro = baseMayor + baseMenor + 2 \* Math.sqrt(Math.pow((baseMayor - baseMenor) / 2, 2) + Mat(altua, 2));
* y por ultimo Se realiza el cálculo del área y el perímetro del trapecio, si todos los valores necesarios están presentes, se realiza el cálculo y se muestra el resultado formateado en tvResultado (un TextView), si alguno de los valores está vacío, se muestra un mensaje indicando al usuario que debe ingresar todos los valores.

Ilustración 11DISEÑO TRAPECIO

# **DISEÑO TRAPECIO**

Dentro de archivo activity.trapecio.xml se inicia un RelativeLayout que define el diseño principal de la actividad. xmlns: android y xmlns: tools son atributos de espacio de nombres para las definiciones de XML.

* Agregamos un TextView que muestra el título de la actividad, el texto, el tamaño de texto y el margen inferior se han configurado.
* Se agrega un RelativeLayout que contiene los elementos específicos para la entrada de datos y botones. Este segundo RelativeLayout tiene un relleno de 30dp.
* Se agrega un EditText con el ID etBaseMayor para que los usuarios ingresen la base mayor del trapecio. Se establece el tipo de entrada como número decimal y se proporciona una sugerencia.
* Se agrega otro EditText con el ID etBaseMenor para la base menor del trapecio.
* Se agrega un tercer EditText con el ID etAltura para que los usuarios ingresen la altura del trapecio.
* Se agrega un botón con el ID btnCalcular que los usuarios pueden hacer clic para realizar el cálculo del área y el perímetro del trapecio.
* Se añade otro TextView con el ID tvResultado que mostrará el resultado del cálculo.
* Se agrega un botón con el ID btnRegresar que permite a los usuarios regresar a la actividad anterior.

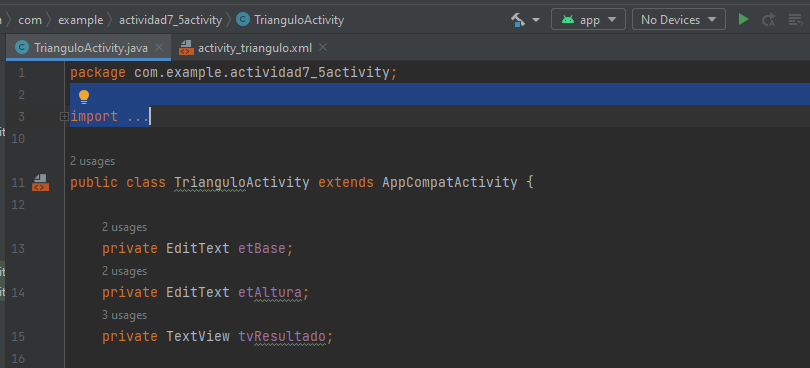


# **Triangulo**

Ilustración 12FIGURA Triangulo

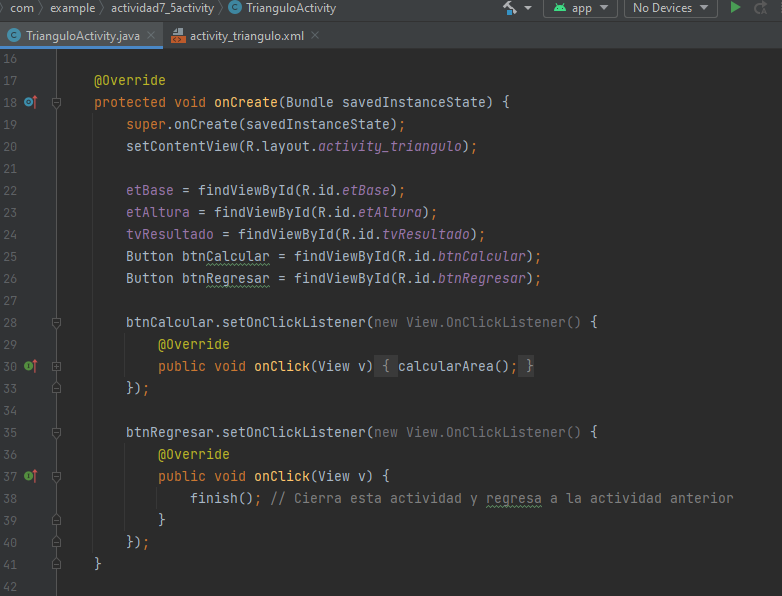
Dentro de este archivó TrianguloActivity la figura tenemos en nombre la clase y principalmente tenemos las importaciones de las clases creadas para que nuestra aplicación funcione correctamente y estas son algunas importaciones que importamos al archivo TrianguloActivity.

* import android.os.Bundle;
* import android.view.View;
* import android.widget.Button;
* import android.widget.EditText;
* import android.widget.TextView;
* import androidx.appcompat.app.AppCompatActivity;
* import java.text.DecimalFormat;



Declaración de la clase TrianguloActivity, que extiende AppCompatActivity.

* Declaración de variables (etBaseMayor, etBase, etAltura y tvResultado) para referenciar los elementos de la interfaz de usuario.
* Iniciamos el método onCreate, que es llamado cuando la actividad se está iniciando. super. onCreate(savedInstanceState) realiza cualquier inicialización necesaria por parte de la clase base, y setContentView(R.layout.activity\_triangulo) establece el contenido de la actividad a partir del archivo de diseño XML llamado activity\_triangulo.xml.
* Se obtienen referencias a los elementos de la interfaz de usuario mediante sus identificadores (findViewById), estos elementos probablemente estén definidos en el archivo de diseño activity\_trianguo.xml.
* Se asigna un listener al botón btnCalcular, cuando se hace clic en este botón, se llama al método calcularArea.
* Se agrega un listener al botón btnRegresar, Cuando se hace clic en este botón, se llama al método finish(), que cierra esta actividad y regresa a la actividad anterior.



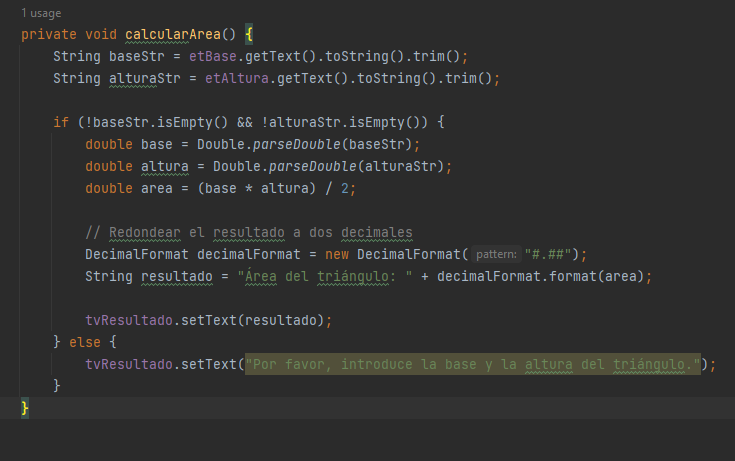
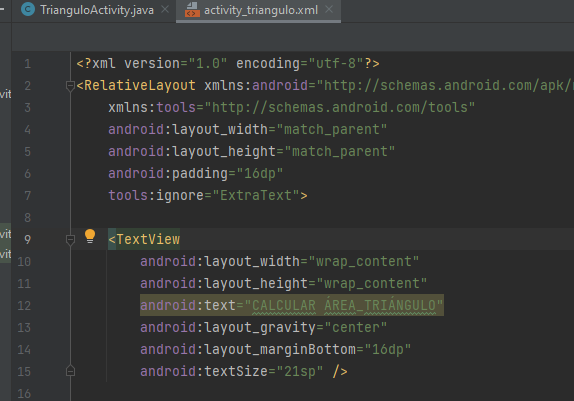
* Definimos el método calcularArea que realiza el cálculo del área del triángulo, y dentro de ellos se crean las operaciones.
* area = (base \* altura) / 2;
* y por ultimo Se realiza el cálculo del área del triángulo, si todos los valores necesarios están presentes, se realiza el cálculo y se muestra el resultado formateado en tvResultado (un TextView), si alguno de los valores está vacío, se muestra un mensaje indicando al usuario que debe ingresar todos los valores.

Ilustración 13DISEÑO TRIANGULO

# **DISEÑO TRAPECIO**

Dentro de archivo activity.triangulo.xml se inicia un RelativeLayout que define el diseño principal de la actividad. xmlns: android y xmlns: tools son atributos de espacio de nombres para las definiciones de XML.



* Agregamos un TextView que muestra el título de la actividad, el texto “CALCULAR AREA\_YTRIANGULO”, el tamaño de texto y el margen inferior se han configurado.
* Se agrega un RelativeLayout que contiene los elementos específicos para la entrada de datos y botones. Este segundo RelativeLayout tiene un relleno de 30dp.
* Se agrega un EditText con el ID etBase para que los usuarios ingresen la base mayor del trapecio. Se establece el tipo de entrada como número decimal y se proporciona una sugerencia.
* Se agrega otro EditText con el ID etAltura para la base menor del trapecio.
* Se agrega un tercer EditText con el ID etAltura para que los usuarios ingresen la altura del trapecio.
* Se agrega un botón con el ID btnCalcular que los usuarios pueden hacer clic para realizar el cálculo del área del triangulo.
* Se añade otro TextView con el ID tvResultado que mostrará el resultado del cálculo.
* Se agrega un botón con el ID btnRegresar que permite a los usuarios regresar a la actividad anterior.



Ilustración 14ARCHIVO DISEÑO MAIN.XML FINAL

# **ARCHIVO DISEÑO MAIN.XML**

