SIMULACIONES

IV + IoT + DL

Industria 5.0

Diego L. Aristizábal Ramírez
Profesor, Facultad de Ciencias, Departamento de Física
Universidad Nacional de Colombia
Medellín

DIBUJANDO
CON ANDROID

4

Temas
✓ INTRODUCCIÓN
✓ MI TERCERA APP: DIBUJANDO PRIMITIVAS
✓ MI CUARTA APP: USA TRANSFORMACIONES AFIN PARA HACER UN GAUGE
✓ DISTRIBUYENDO LA APP
✓ TAREA

INTRODUCCIÓN

En este módulo se desarrollarán dos aplicaciones que ilustran conceptos básicos de cómo hacer dibujos con ANDROID. En la primera app (**MiTerceraApp** del curso) se ilustra cómo realizar dibujos de las denominadas formas primitivas (líneas, círculos, rectángulos…), pero sin tener el cuidado que sean **responsivos**, es decir, que se vean de la misma forma y proporcionalidad en todos los dispositivos móviles, independientemente de las dimensiones, resoluciones, razón de aspecto, o si son celulares, tabletas o smartwatch. En la segunda app (**MiCuartaApp** del curso) se abordan dos de las denominadas transformaciones afines (traslación y rotación) que facilita el dibujo de un **GAUGE**. Adicionalmente este GAUGE se diseña con **responsividad**.

MI TERCERA APP: DIBUJANDO PRIMITIVAS

Los requisitos y el diseño

Esta aplicación se denominará **MiTerceraApp** y la conforman una actividad principal denominada **ActividadPrincipalMiTerceraApp** (clase que hereda de **Activity**) y de un paquete que se denominara **componentes** que contiene una clase que **hereda** de **View** y que se denomina **Pizarra**. Pizarra al **heredar** de **View** permite realizar dibujos de las formas primitivas.

La implementación

Paso 1

Ejecutar **Android Studio**. Se despliega la interfaz de la Figura 1.

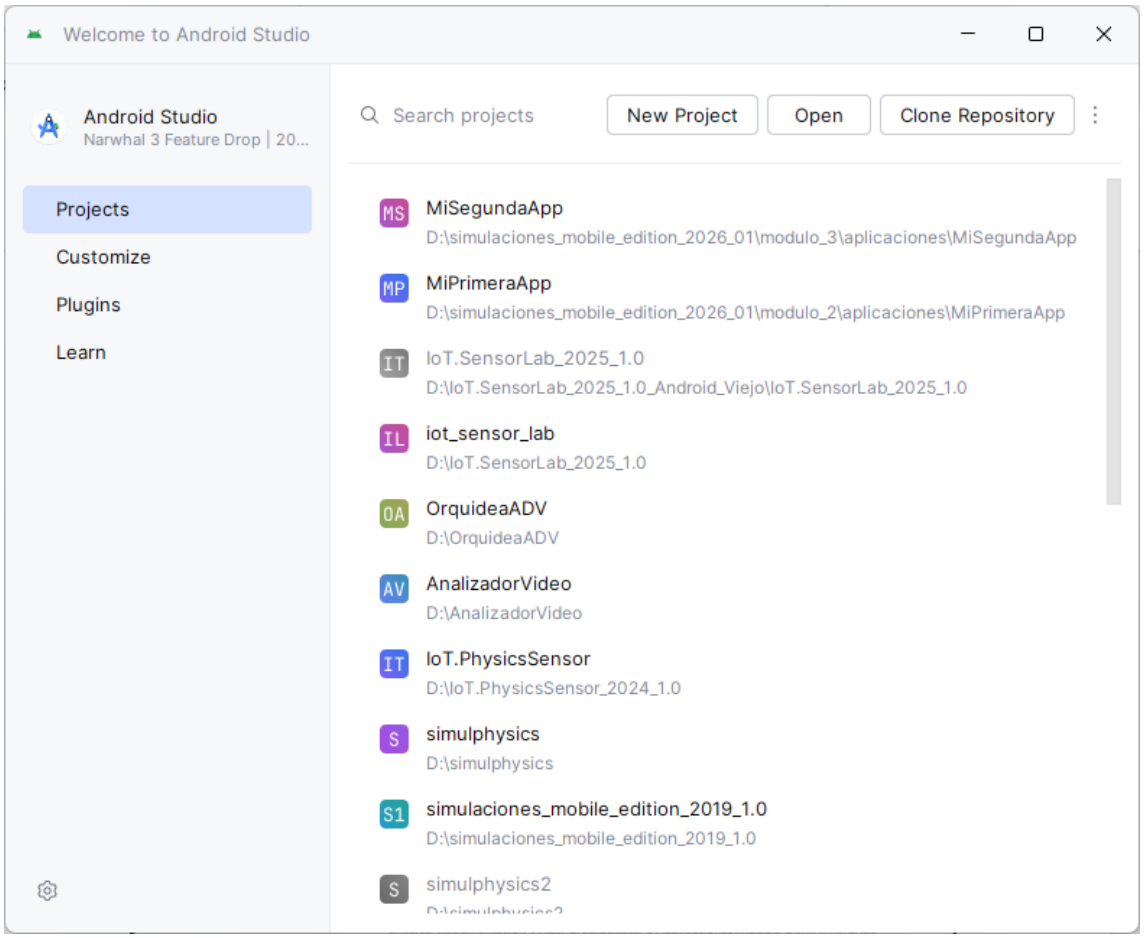

Figura 1

Paso 2:

Hacer clic en **New Project** para iniciar el proyecto. Se despliega la interfaz de la Figura 2.

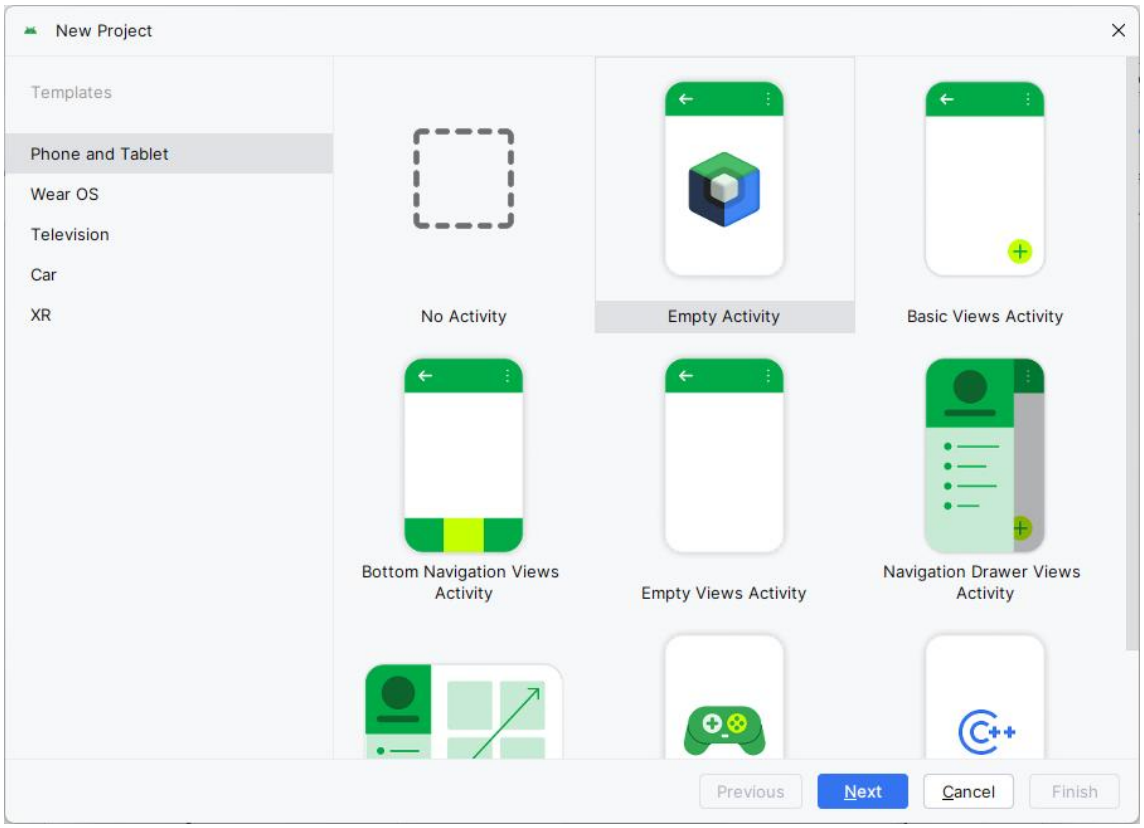

Figura 2

Paso 3:

Seleccionar **No Activity** y hacer clic en el botón **Next** lo cual desplegará la GUI de la Figura 3. Llenar los campos respectivos. El nombre de la aplicación debe comenzar con mayúsculas. En este caso se le denominó **MiTerceraApp**. Al dominio de la compañía se le denominó **com.curso_simulaciones**. Llenar el campo que elige el directorio donde se hospedará la aplicación. Luego el campo del lenguaje, en nuestro caso **Java**. Luego elegir la mínima Api que soportará la app, en nuestro caso elegimos la API 24 que nos garantiza que se nuestra app se ejecutará aproximadamente en el 98.6 % de los dispositivos ANDROID actuales. Hacer clic en **Finish** y se despliega la interfaz de la Figura 4.

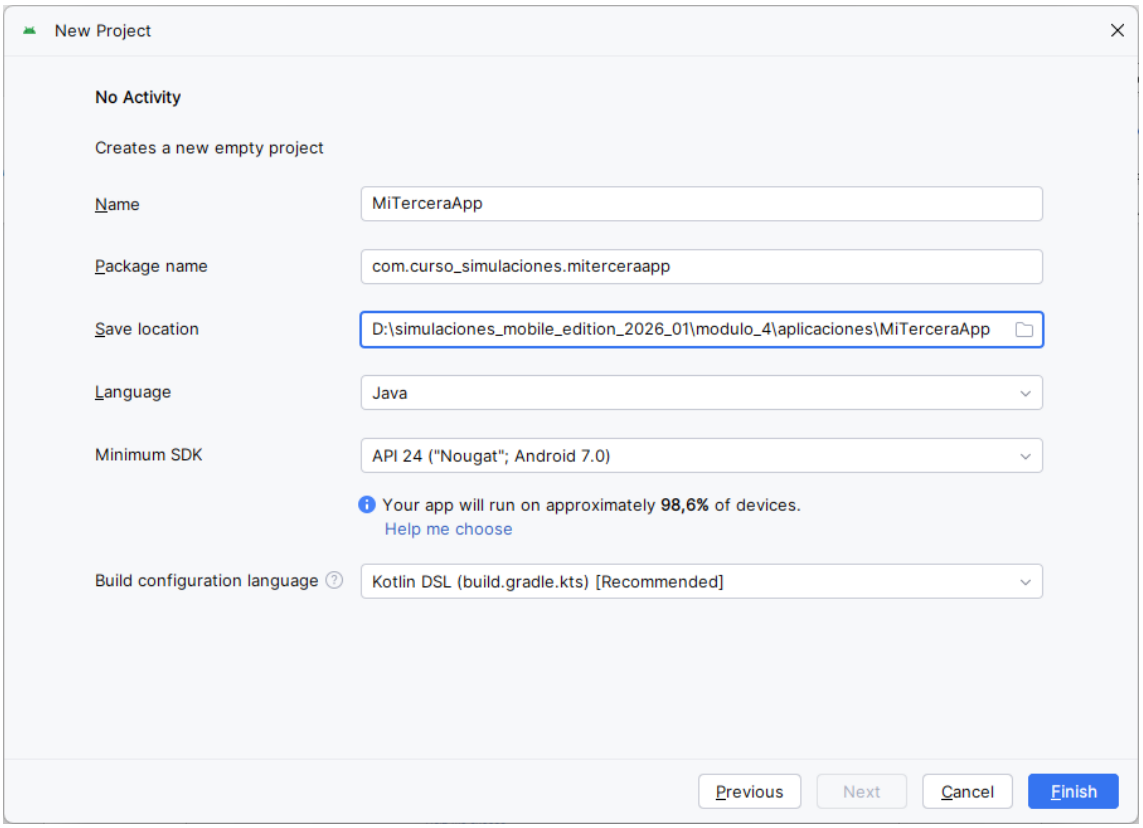

Figura 3

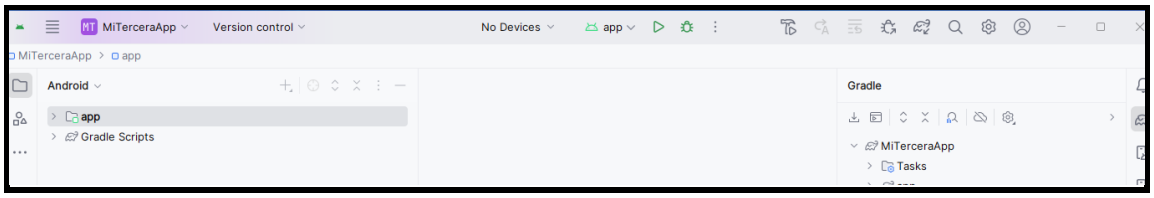

Figura 4

Paso 4:

Ubicarse sobre **java/com/curso_simulaciones>miterceraapp**, y hacer clic derecho. Se despliega la interfaz de la Figura 5,

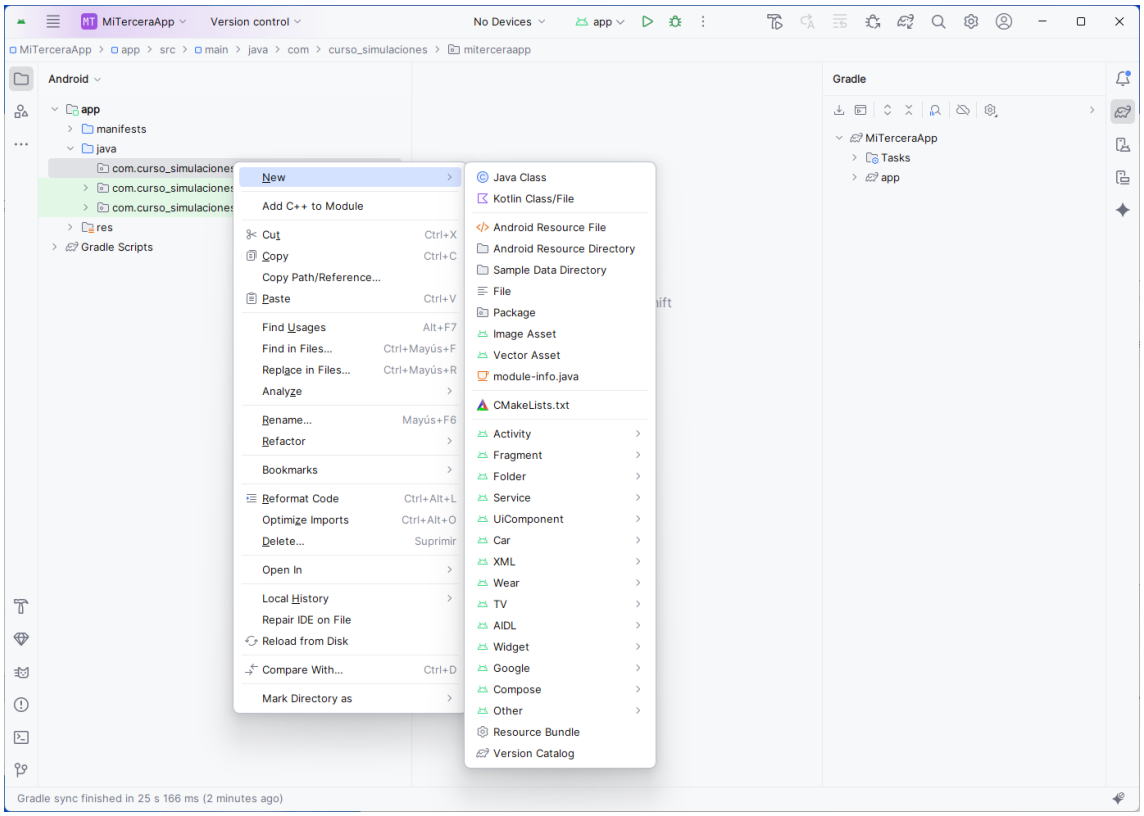

Figura 5

Seleccionar **New > Java Class**. Se despliega la interfaz de la Figura 6. En el campo escribir el nombre de la clase que se creará, en este caso se eligió el nombre **ActividadPrincipalMiTerceraApp**. Hacer doble clic **Class**. Se despliega la interfaz de la Figura 7.

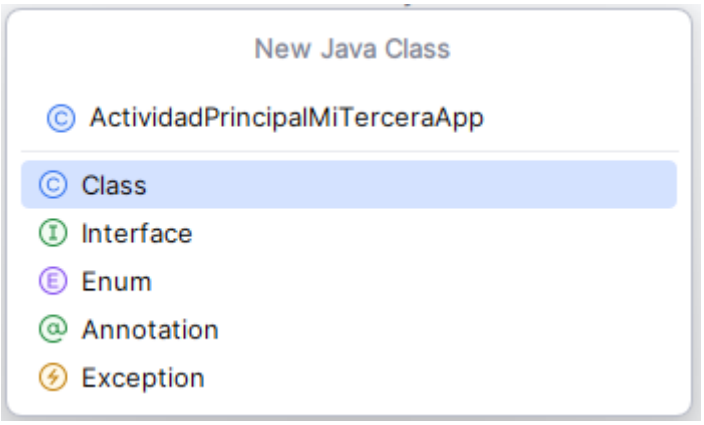

Figura 6

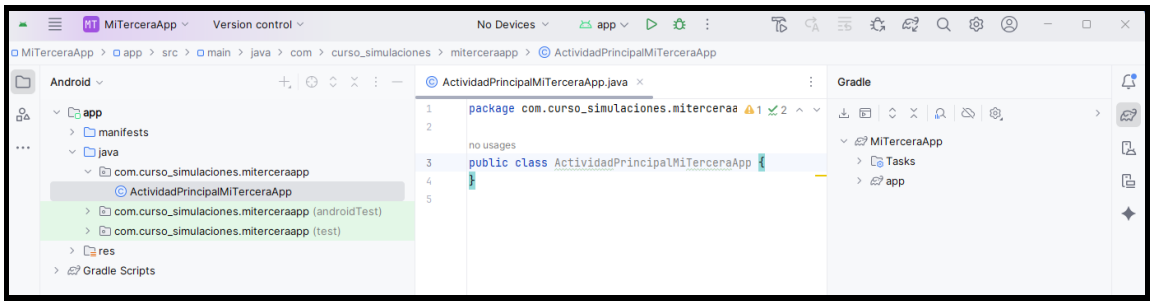

Figura 7

Paso 5:

Agregar el siguiente código, Figura 8:

```
extends Activity
```

Importar los recursos necesarios de la API de Android que contenga la clase **Activity**, Figura 8. Esto haciendo clic en la tecla **Enter** mientras se presiona simultáneamente la tecla **Alt**. Hecho esto **Activity** cambia de color rojo a gris, Figura 9.

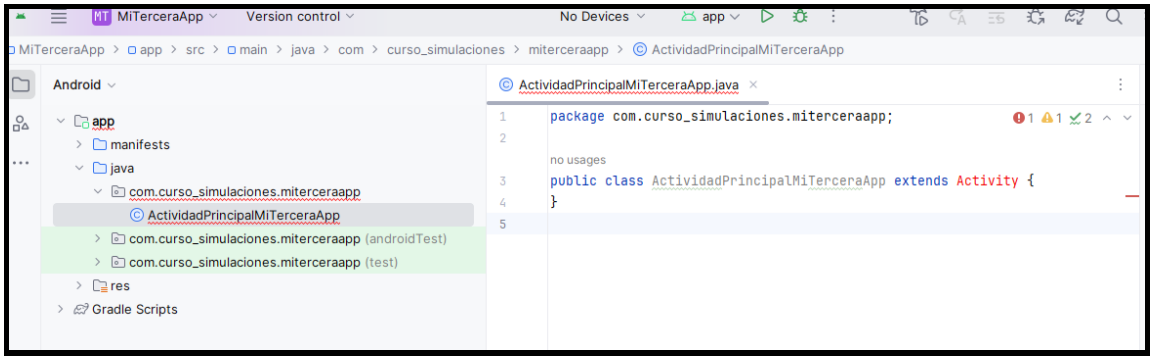

Figura 8

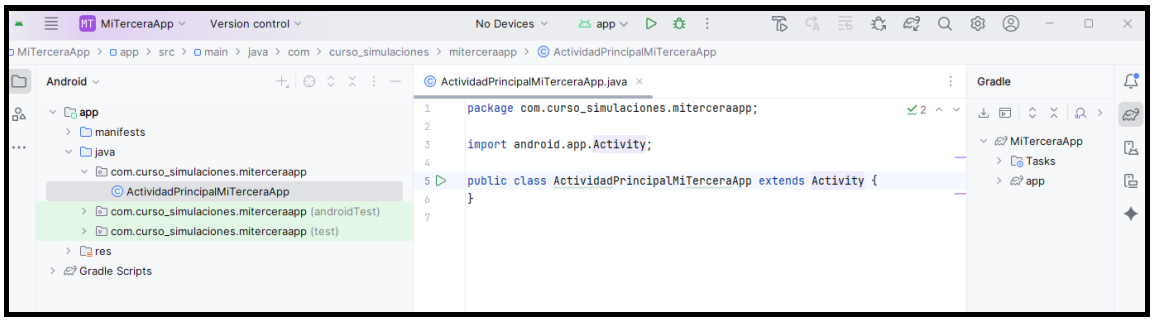

Figura 9

Antes de continuar implementando la clase **ActividadPrincipalMiTerceraApp** se creará el paquete componentes y su clase **Pizarra**.

Paso 6:

Ubicarse sobre **java/com/curso_simulaciones>miterceraapp**, y hacer clic derecho. Se despliega la interfaz de la Figura 10.

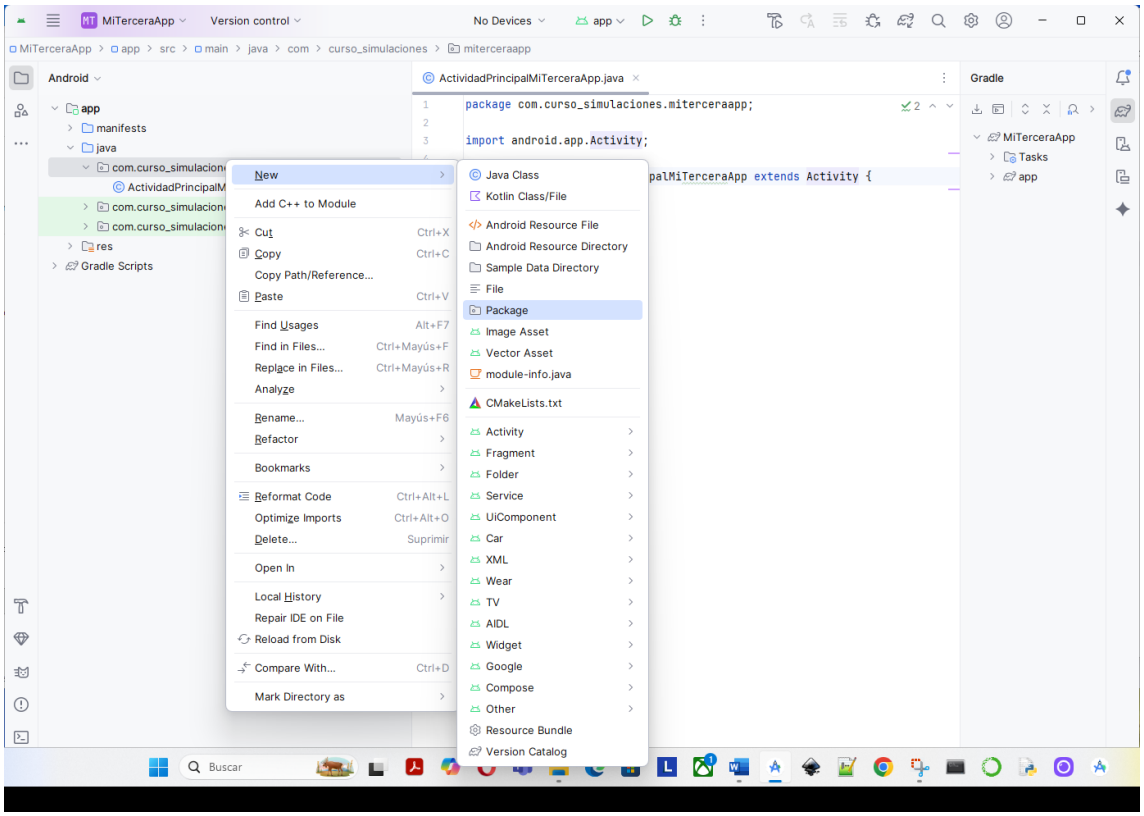

Figura 10

Seleccionar **New > Package**. Se despliega la interfaz de la Figura 11. En el campo agregar el nombre del paquete, **componentes**. Hacer **ENTER**. Se despliega la interfaz de la Figura 12.

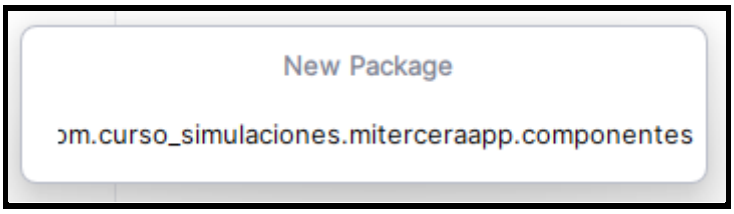

Figura 11

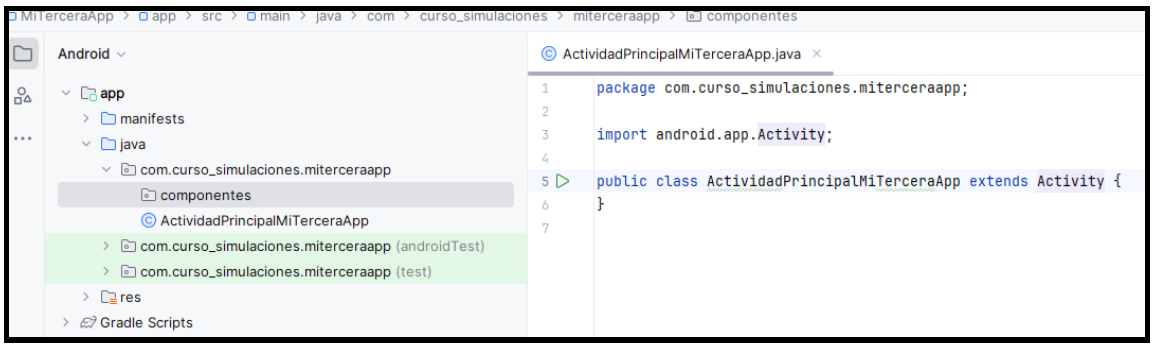

Figura 12

Ubicar el cursor en **componentes** hacer clic derecho, seleccionar **New > Java Class** y en el campo escribir el nombre de la clase, **Pizarra**. Hacer doble clic en **Class**. La GUI se transforma en la de la Figura 13.

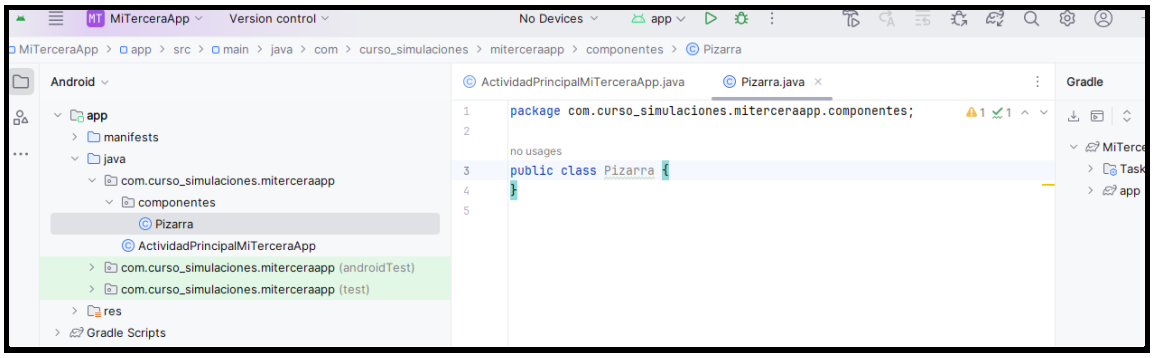

Figura 13

Paso 7:

Agregar código a la clase **ActividadPrincipalMiTerceraApp**

Agregar el siguiente código a la clase **ActividadPrincipalMiTerceraApp**. Importar los recursos en las clases que aparezcan en rojo tal como se ha venido haciendo (…clic en la tecla **Enter** mientras se presiona simultáneamente la tecla **Alt**).

```
public class ActividadPrincipalMiTerceraApp extends Activity {


    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

    /*llamada al método para crear los elementos de la interfaz
    gráfica de usuario (GUI)*/
        crearElementosGui();


    /*para informar cómo se debe adaptar la GUI a la pantalla del
    dispositivo*/
        ViewGroup.LayoutParams parametro_layout_principal = new
ViewGroup.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT,
ViewGroup.LayoutParams.MATCH_PARENT);


    /*pegar al contenedor la GUI: en el argumento se está llamando
    al método crearGui()*/
        this.setContentView(crearGui(), parametro_layout_principal);


    }


    /*crear los objetos de la interfaz gráfica de usuario (GUI)*/
    private void crearElementosGui() {


    }


    /*organizar la distribución de los objetos de de la GUI usando
    administradores de diseño*/
    private LinearLayout crearGui() {

        //administrador de diseño
        LinearLayout linear_principal = new LinearLayout(this);
        linear_principal.setOrientation(LinearLayout.VERTICAL);
        linear_principal.setGravity(Gravity.CENTER_HORIZONTAL);
        linear_principal.setGravity(Gravity.FILL);
        linear_principal.setBackgroundColor(Color.rgb(250, 150, 50));


        return linear_principal;

    }

}
```


Paso 8: El archivo manifiesto

Hay que recordar que toda `Activity` se debe declarar en el archivo manifiesto. Ir al **`AndroidManifest.xml`**.

Ahora es necesario agregar código en el archivo manifiesto. Hacer clic en el archivo **app.manifests> AndroidManifest.xml**. En el código desplegado agregar código de tal forma que el archivo quede como el siguiente.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.curso_simulaciones.miterceraapp">

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.MiTerceraApp"
        tools:targetApi="31" >

    <activity
android:name="com.curso_simulaciones.miterceraapp.ActividadPrincipalMiTerceraApp"
        android:exported="true">

        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>

    </application>


</manifest>
```

Ya se está en condiciones de proceder a **compilar** y **ejecutar** la aplicación.

Paso 9:

Conectar el dispositivo por un puerto USB del PC. Proceder a la **compilación y ejecución** haciendo clic en el botón de la barra superior de herramientas de Android Studio que está señalado con un círculo rojo en la Figura 14. Se despliega la GUI de la Figura 15.

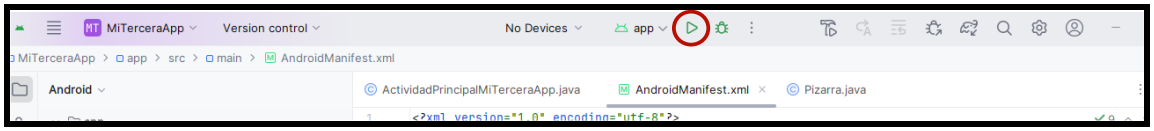

Figura 14

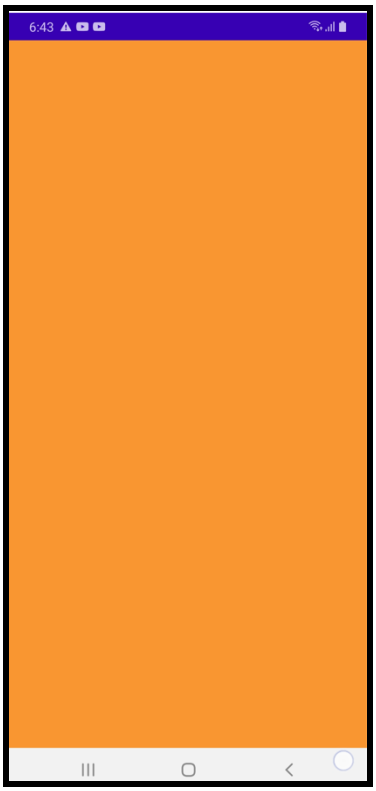

Figura 15

Paso 10: Crear el código inicial de la clase Pizarra

Agregar el siguiente código a la clase **Pizarra**.

```
public class Pizarra extends View {

    //constructor
    public Pizarra(Context context) {
        super(context);
    }

}
```

Importar los recursos necesarios de la API de Android que contenga la clase **View** y **Context**. Ubicando el cursor sobre **View** hacer clic en la tecla **Enter** mientras se presiona simultáneamente la tecla **Alt**. Repetir para **Context**.

Paso 11: Agregar más código a la clase **ActividadPrincipalMiTerceraApp**

A continuación, se agrega código a la clase **ActividadPrincipalMiTerceraApp**. Está remarcado en amarillo.

```
public class ActividadPrincipalMiTerceraApp extends Activity {

    private Pizarra lienzo;

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        /*llamada al método para crear los elementos de la interfaz
        gráfica de usuario (GUI)*/
        crearElementosGui();


        /*para informar cómo se debe adaptar la GUI a la pantalla del
        dispositivo*/
        ViewGroup.LayoutParams parametro_layout_principal = new
                ViewGroup.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT,
                ViewGroup.LayoutParams.MATCH_PARENT);


        /*pegar al contenedor la GUI: en el argumento se está llamando
        al método crearGui()*/
        this.setContentView(crearGui(), parametro_layout_principal);


    }


    /*crear los objetos de la interfaz gráfica de usuario (GUI)*/
    private void crearElementosGui(){

        //crear objeto Pizarra
        lienzo=new Pizarra(this);
        //darle color blanco al lienzo antes de pegar
        lienzo.setBackgroundColor(Color.WHITE);



    }


    /*organizar la distribución de los objetos de de la GUI usando
    administradores de diseño*/
    private LinearLayout crearGui(){

        //administrador de diseño
        LinearLayout linear_principal = new LinearLayout(this);
        linear_principal.setOrientation(LinearLayout.VERTICAL);
        linear_principal.setGravity(Gravity.CENTER_HORIZONTAL);
        linear_principal.setGravity(Gravity.FILL);
        linear_principal.setBackgroundColor(Color.rgb(250,150,50));



        /*parametro para pegar el lienzo*/
        LinearLayout.LayoutParams parametrosPegadaLienzo= new
LinearLayout.LayoutParams(android.view.ViewGroup.LayoutParams.MATCH_PARENT,0);
```

```
        parametrosPegadaLienzo.setMargins(20, 20, 20, 20);
        parametrosPegadaLienzo.weight = 1.0f;

        //pegar lienzo
        linear_principal.addView(lienzo,parametrosPegadaLienzo);



        return linear_principal;

    }



}
```


Conectado su dispositivo móvil vía USB al computador proceder a la **compilación** y **ejecución** haciendo clic en el botón de la barra superior de herramientas de **Android Studio** que está señalado con un círculo rojo en la Figura 16. Se despliega la GUI de la Figura 17.

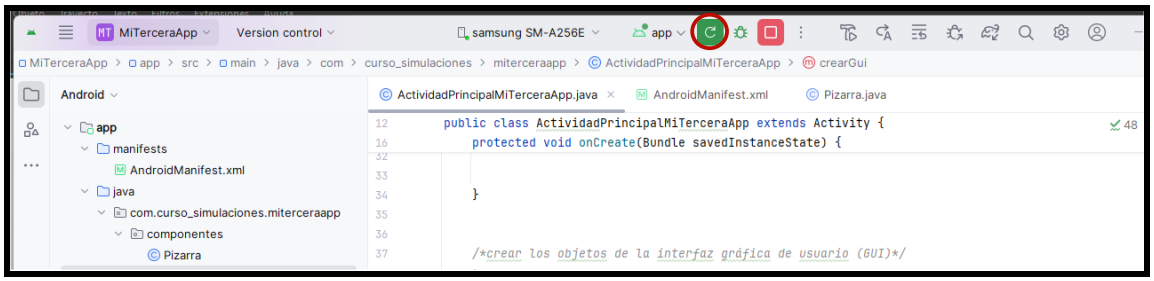

Figura 16

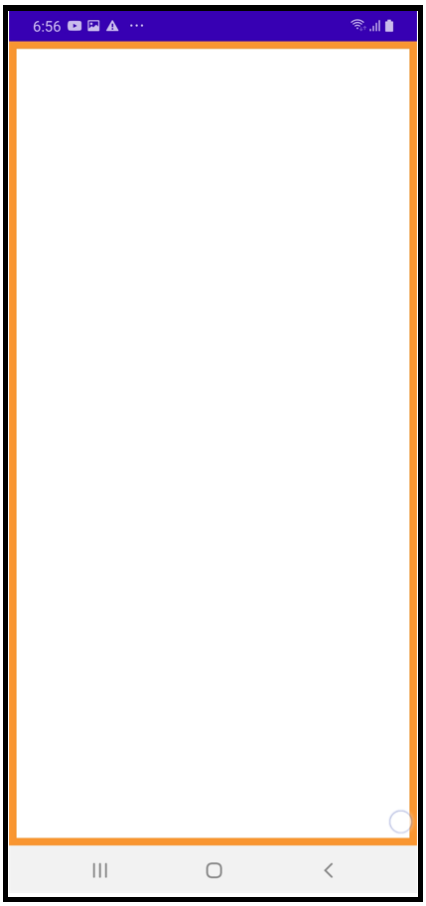

Figura 17

Ya se tiene el lienzo para comenzar a realizar los dibujos sobre éste. Para esto se agregará código a la clase Pizarra.

Paso 12: Dibujando una cadena (**String**)

Agregar en la clase Pizarra el método **onDraw( )** con el siguiente código:

```
public class Pizarra extends View {


    //constructor
    public Pizarra(Context context) {
        super(context);
    }

    //método para dibujar
    protected void onDraw(Canvas canvas) {

        Paint pincel = new Paint();
        pincel.setAntiAlias(true);

        /*Tamaño del texto*/
        pincel.setColor(Color.RED);
        pincel.setTextSize(40f);
        /*texto que comienza a escribirse en (20,30) de tamaño de letra 40 pixeles y de
color rojo */
```

```
        canvas.drawText("Hola Jóvenes, bienvenidos a sus primeros dibujos", 20f, 50f,
pincel);

    }

}
```

Importar los recursos de las clases que aparezcan en rojo.

Compilar y **ejecutar** la aplicación, se despliega la GUI de la Figura 18.

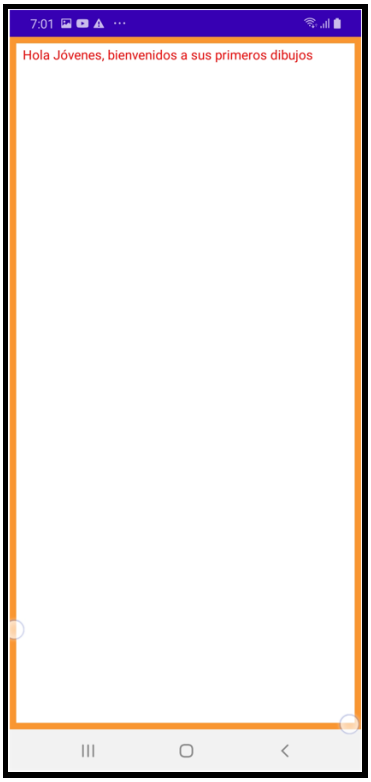

Figura 18

Se puede observar que si se rota el dispositivo móvil rota la GUI. Si se quiere que no suceda esto, es decir, que permanezca horizontal o vertical al rotar la pantalla se debe dar esta orden en el archivo manifiesto. En esta app se quiere que se mantenga la GUI horizontal así se rote la pantalla. Para esto agregar al archivo manifiesto el código remarcada en amarillo.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.curso_simulaciones.miterceraapp">

    <application
        android:allowBackup="true"
```

```
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.MiTerceraApp" >

        <activity
android:name="com.curso_simulaciones.miterceraapp.ActividadPrincipalMiTerceraApp"
            android:exported="true"
            android:screenOrientation="landscape">

            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

    </application>



</manifest>
```


Compilar y **ejecutar**. Se despliega la GUI de la Figura 19.

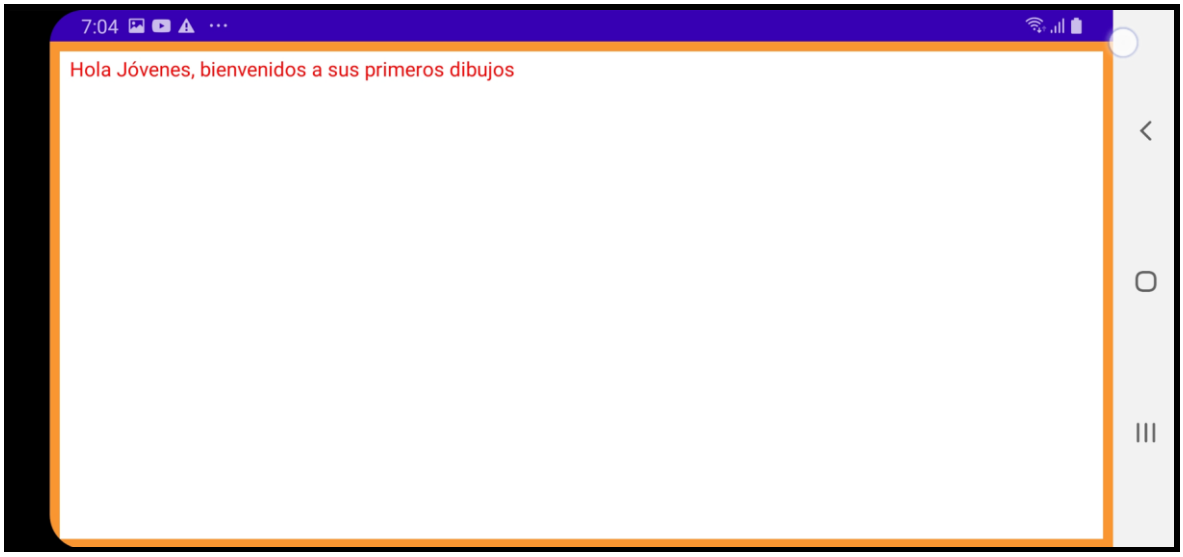

Figura 19

Paso 13: **Dibujar una línea**

Dentro del método **onDraw( )** de la clase **Pizarra** agregar el siguiente código:

```
/*
Dibuja un línea negra de espesor 5
desde (30,100) hasta (200,180)
*/
pincel.setStrokeWidth(5);
pincel.setColor(Color.BLACK);
canvas.drawLine(30, 100, 200,180, pincel);
//método para dibujar
protected void onDraw(Canvas canvas) {

Paint pincel = new Paint();
pincel.setAntiAlias(true);

/*Tamaño del texto*/
pincel.setColor(Color.RED);
pincel.setTextSize(40f);

/*texto que comienza a escribirse en (20,30) de tamaño de letra 40 pixeles y de color
rojo */

canvas.drawText("Hola Jóvenes, bienvenidos a sus primeros dibujos", 20f, 50f, pincel);

/*
Dibuja un línea negra de espesor 5
desde (30,100) hasta (200,180)
*/
pincel.setStrokeWidth(5);
pincel.setColor(Color.BLACK);
canvas.drawLine(30, 100, 200, 180, pincel);


}
```


Compilar y **ejecutar** la aplicación, se despliega la GUI de la Figura 20.

Figura 20

Paso 14: Dibujando un círculo

Agregar en el método onDraw ( ) de la clase Pizarra el código,

```
/*
  Dibujar círculo de color rojo no
  relleno con centro en (600, 150) y
  radio 150 pixeles. Grosor de trazo 10
*/
pincel.setStrokeWidth(10);
pincel.setStyle(Paint.Style.STROKE);
pincel.setColor(Color.RED);
canvas.drawCircle(600, 350, 150, pincel);
```



```
//método para dibujar
protected void onDraw(Canvas canvas) {

    Paint pincel = new Paint();
    pincel.setAntiAlias(true);

    /*Tamaño del texto*/
    pincel.setColor(Color.RED);
    pincel.setTextSize(40f);
    /*texto que comienza a escribirse en (20,30) de tamaño de letra 40 pixeles y de
color rojo */
    canvas.drawText("Hola Mundo", 20f, 50f, pincel);


    /*
    Dibuja un línea negra de espesor 5
    desde (30,100) hasta (200,180)
    */
    pincel.setStrokeWidth(5);
    pincel.setColor(Color.BLACK);
    canvas.drawLine(30, 100, 200,180, pincel);


    /*
    Dibujar círculo de color rojo no
    relleno con centro en (600, 150) y
    radio 150 pixeles. Grosor de trazo 10
    */
    pincel.setStrokeWidth(10);
    pincel.setStyle(Paint.Style.STROKE);
    pincel.setColor(Color.RED);
    canvas.drawCircle(600, 350, 150, pincel);


}
```

Compilar y ejecutar la aplicación, se despliega la GUI de la Figura 21.

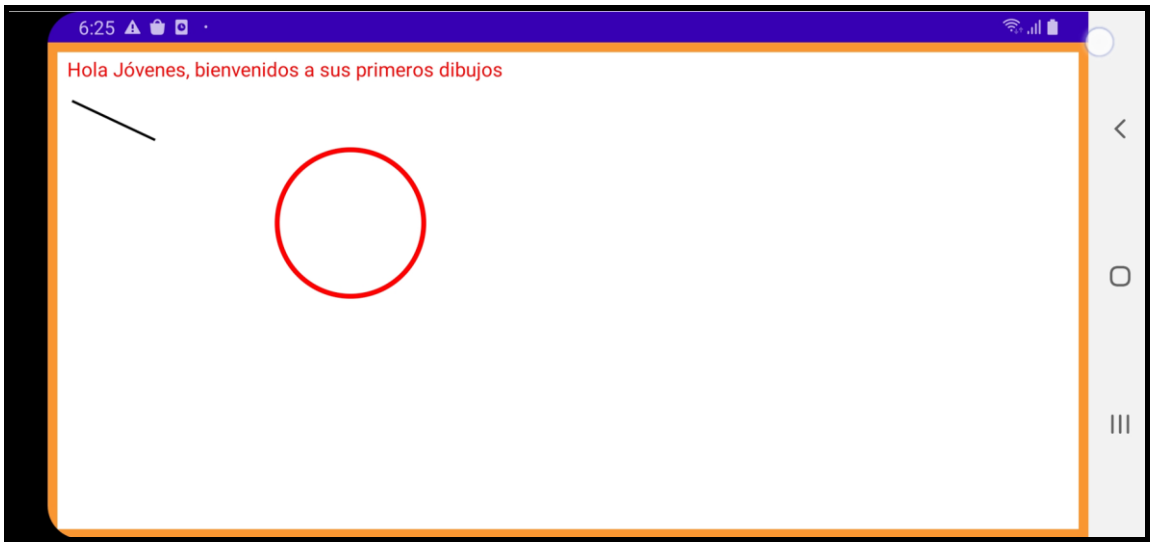

Figura 21

Paso 15: Dibujando un rectángulo relleno y transparente

Agregar en el método `onDraw` de Pizarra el código,

```
/*
 Dibujar rectángulo de color verde
 r=0, g=250, b=0) con 100 de
 transparencia relleno con esquina
 superior izquierda en (400,150) y
 esquina inferior derecha en (800,550)
*/
pincel.setStyle(Paint.Style.FILL);
pincel.setColor(Color.argb(100, 0, 250, 0));
canvas.drawRect(400, 150, 800, 550, pincel);
```

Compilar la aplicación y ejecutarla. Se desplegará en el dispositivo la pantalla de la Figura 22.

Figura 22

Paso 16: Dibujando un rectángulo relleno y opaco

Agregar en el método onDraw de Pizarra el código,

```
/*
 Dibujar rectángulo de color rojo
 r=255, g=0, b=0) con esquina
 superior izquierda en (350,250) y
 esquina inferior derecha en (850,450)
*/
pincel.setStyle(Paint.Style.FILL);
pincel.setColor(Color.rgb(255, 0, 50));
canvas.drawRect(350, 250, 850, 450, pincel);
```

Compilar la aplicación y ejecutarla. Se desplegará en el dispositivo la pantalla de la Figura 23.

Figura 23

Paso 17: Dibujando un rectángulo sin relleno

Agregar en el método onDraw de Pizarra el código,

```
/*
 Dibujar rectángulo cuyo
 borde es negro de espesor 2,
 no relleno, con esquina superior
 izquierda en (550,100) y
 esquina inferior derecha en (700,200)
*/
pincel.setStyle(Paint.Style.STROKE);
pincel.setStrokeWidth(2);
pincel.setColor(Color.BLACK);
canvas.drawRect(550, 100, 700,200, pincel);
```

Compilar la aplicación y ejecutarla. Se desplegará en el dispositivo la pantalla de la Figura 24.

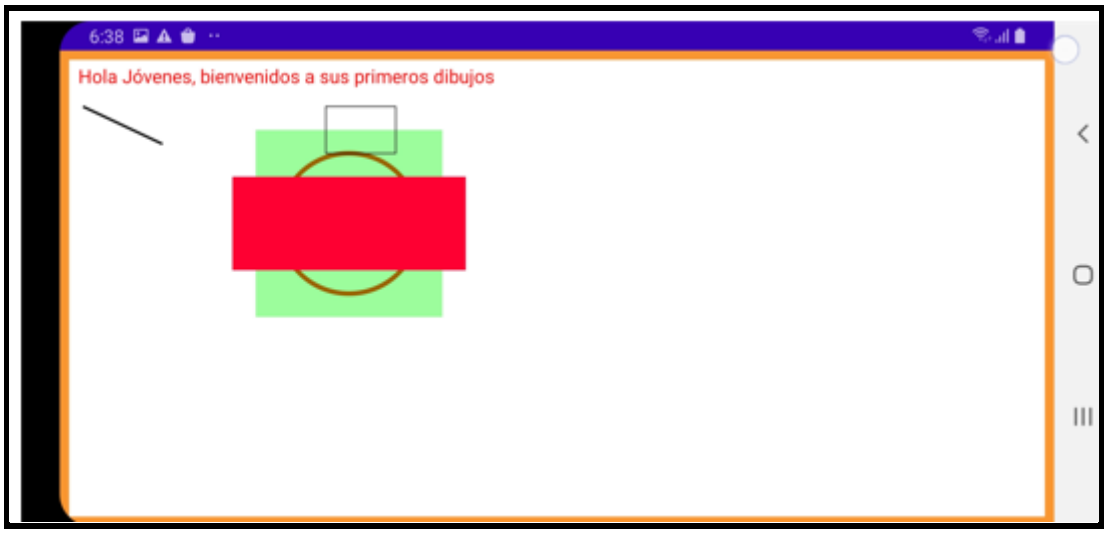

Figura 24

Paso 18: Dibujando un arco

Agregar en el método onDraw de Pizarra el código,

```
/*
 Dibujar arco inscrito en rectángulo
 con esquina superior izquierda en (50,200)
 y esquina inferior derecha en (300,450),
 con inicio en 90 grados y desplazándose
 135 grados (el arco es de 135 grados).
 Color magenta y espesor de línea 2.
*/
RectF rectF = new RectF(50, 200, 300, 450);
pincel.setColor(Color.MAGENTA);
canvas.drawArc(rectF, 90, 135, false, pincel);
```

Compilar la aplicación y ejecutarla. Se desplegará en el dispositivo la pantalla de la Figura 25.

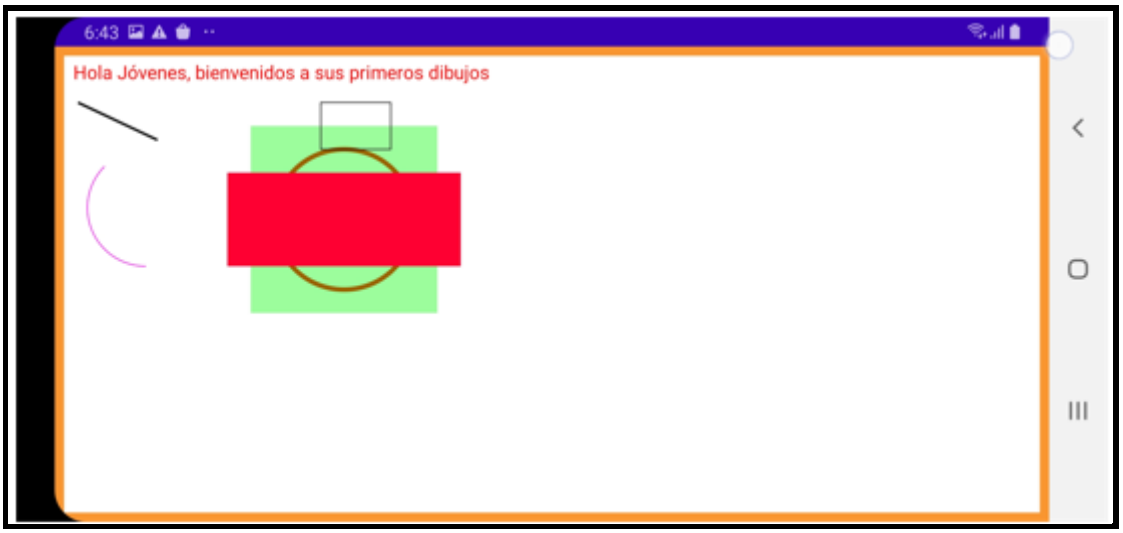

Figura 25

Paso 19: Dibujando sector circular no relleno

Agregar en el método onDraw de Pizarra el código,

```
/*
 Dibujar sector circular inscrito en rectángulo
 con esquina superior izquierda en (50,400)
 y esquina inferior derecha en (300,650),
 no relleno, de espesor de línea 0.5 y de color negro,
 con inicio en 90 grados y desplazándose
 135 grados (el arco es de 135 grados).
*/
RectF rectF_1 = new RectF(50, 400, 300, 650);
pincel.setStrokeWidth(0.5f);
pincel.setColor(Color.BLACK);
canvas.drawArc(rectF_1, 90, 135, true, pincel);
```

Compilar la aplicación y ejecutarla. Se desplegará en el dispositivo la pantalla de la Figura 26.

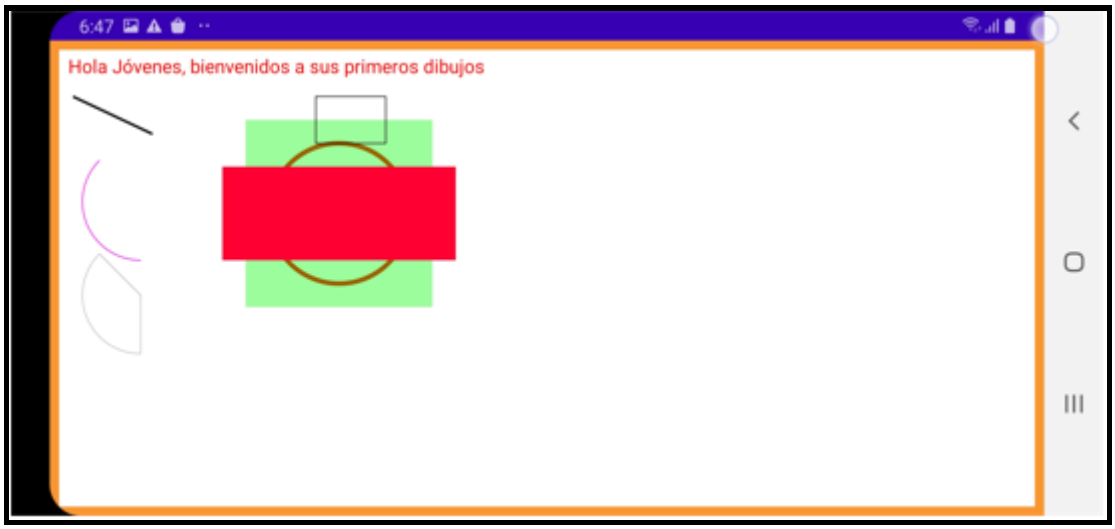

Figura 26

Paso 20: Dibujando sector circular relleno

Agregar en el método onDraw de Pizarra el código,

```
/*
 Dibujar sector circular inscrito en rectángulo
 con esquina superior izquierda en (250,400)
 y esquina inferior derecha en (500,650),
 relleno, de color negro con inicio en 90 grados
 y desplazándose 135 grados (el arco es de 135 grados).
*/
RectF rectF_2 = new RectF(250, 400, 500, 650);
pincel.setStyle(Paint.Style.FILL);
canvas.drawArc(rectF_2, 90, 135, true, pincel);
```

Compilar la aplicación y ejecutarla. Se desplegará en el dispositivo la pantalla de la Figura 27.

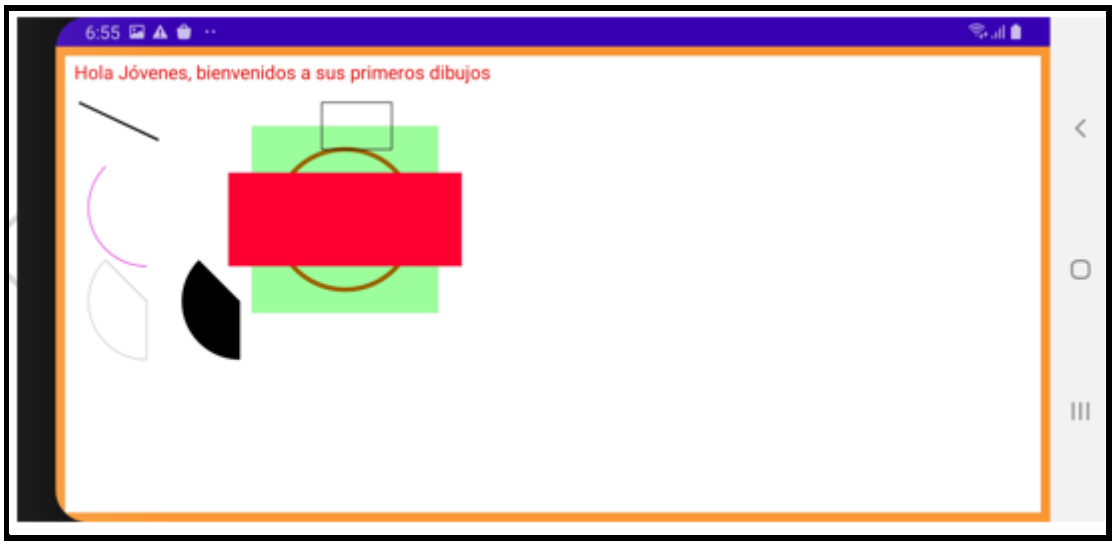

Figura 27

Paso 21: Dibujando triángulo no relleno

Agregar en el método onDraw de Pizarra el código,

```
/*
 Dibujar triángulo con línea azul y vértices en
(600,400); (300,650); (100,50)
*/
pincel.setStyle(Paint.Style.STROKE);
pincel.setStrokeWidth(2);
pincel.setColor(Color.BLUE);

Point a = new Point(600, 400);
Point b = new Point(300, 650);
Point c = new Point(700, 500);

Path path = new Path();
path.moveTo(a.x, a.y);
path.lineTo(b.x, b.y);
path.moveTo(b.x, b.y);
path.lineTo(c.x, c.y);
path.moveTo(c.x, c.y);
path.lineTo(a.x, a.y);
path.close();

canvas.drawPath(path, pincel);
```

Compilar la aplicación y ejecutarla. Se desplegará en el dispositivo la pantalla de la Figura 28.

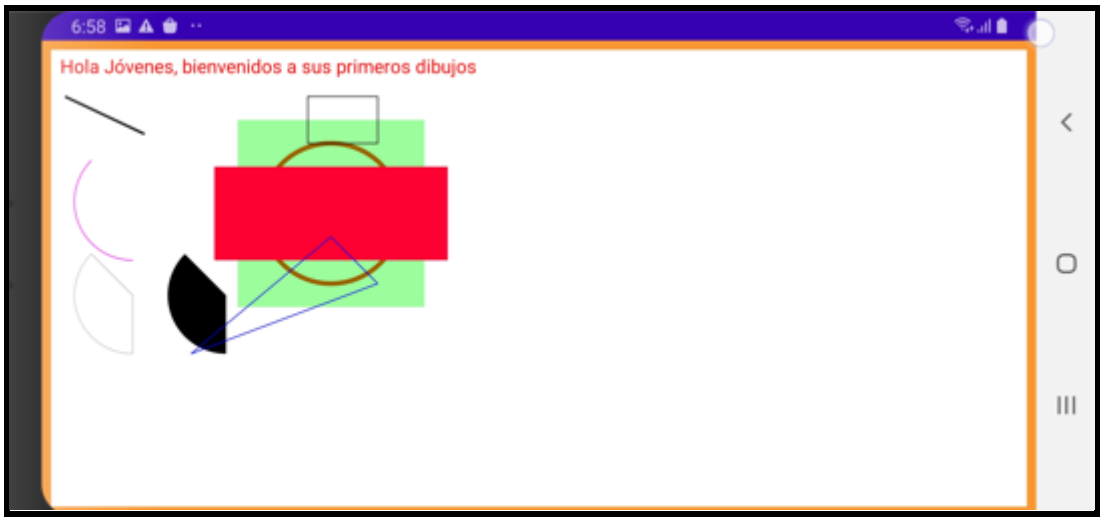

Figura 28

Paso 22: Dibujando algo especial

Agregar en el método onDraw de Pizarra el código,

```
/*
Algo especial.
Muchísimos efectos se pueden lograr con las
librerías de dibujo de ANDROID, sin embargo
no es objeto de estas notas. Aquí con lo básico
es suficiente.
 */
Path trazo = new Path();
trazo.addCircle(800, 400, 100, Path.Direction.CCW);
pincel.setColor(Color.CYAN);
pincel.setStrokeWidth(8);
pincel.setStyle(Paint.Style.STROKE);
canvas.drawPath(trazo, pincel);
pincel.setStrokeWidth(1);
pincel.setStyle(Paint.Style.FILL);
pincel.setTextSize(20);
pincel.setTypeface(Typeface.SANS_SERIF);
pincel.setColor(Color.BLACK);

canvas.drawTextOnPath("Escuela de Física Universidad Nacional de Colombia  -Medellín-",
trazo, 10, 40, pincel);
```

Compilar la aplicación y ejecutarla. Se desplegará en el dispositivo la pantalla de la Figura 29.

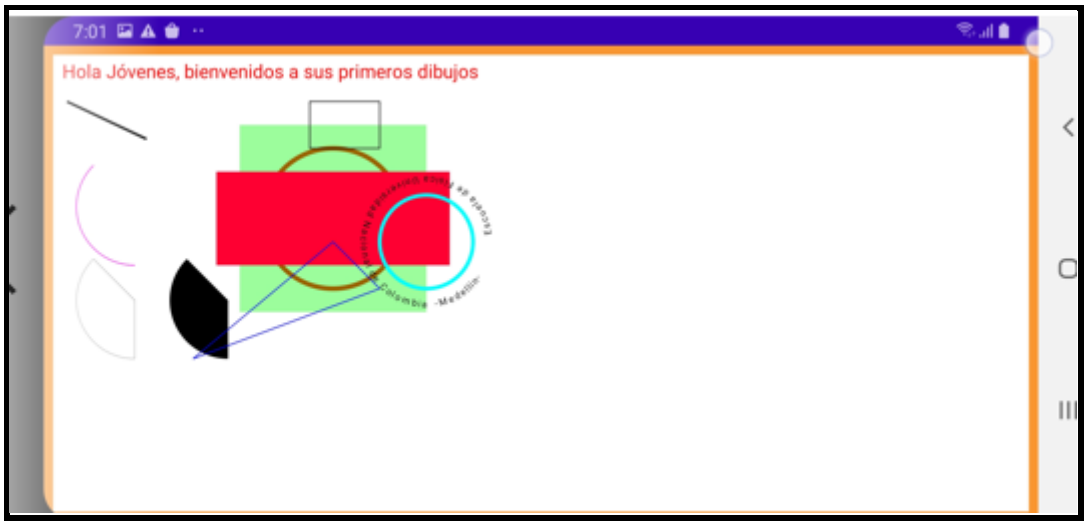

Figura 29

MI CUARTA APP: MI PRIMERA CLASE PARA GENERAR OBJETOS GAUGES

Los requisitos y el diseño

Esta aplicación se denominará **MiCuartaApp** y la conforman una actividad principal denominada **ActividadPrincipalMiCuataApp** (clase que hereda de **Activity**) y de un paquete que se denominara **componentes** que contiene una clase que se denomina **GaugeSimple** que hereda de **View,** lo cual permite generar con ella **instancias (u objetos)** dibujables.

La app desplegará **tres instancias (u objetos)** de la clase **GaugeSimple** tal y como se ilustar en la Figura 30. A estas **instancias (u objetos)** se les creó con variaciones en sus **propiedades (o atributos)**.

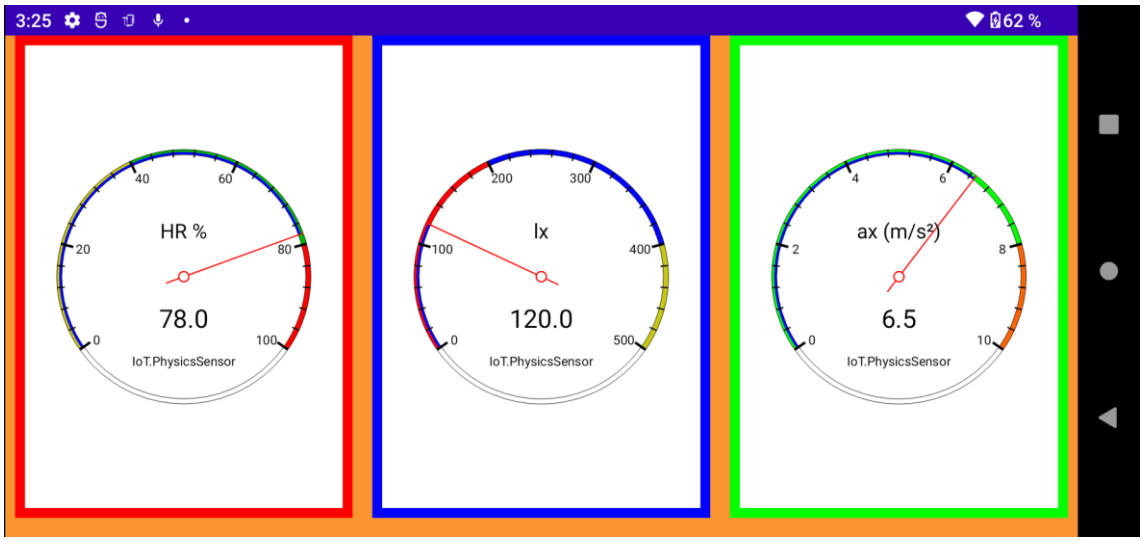

Figura 30

La jerarquía de clases y paquetes es la ilustrada en la Figura 31 que la componen la clase **ActividadPrincipalMiCuartaApp** y el paquete **componentes** que contiene la clase **GaugeSimple**.

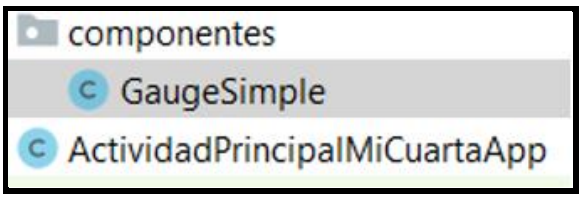

Figura 31

La implementación

Paso 1

Ejecutar **Android Studio**. Se despliega la interfaz de la Figura 32.

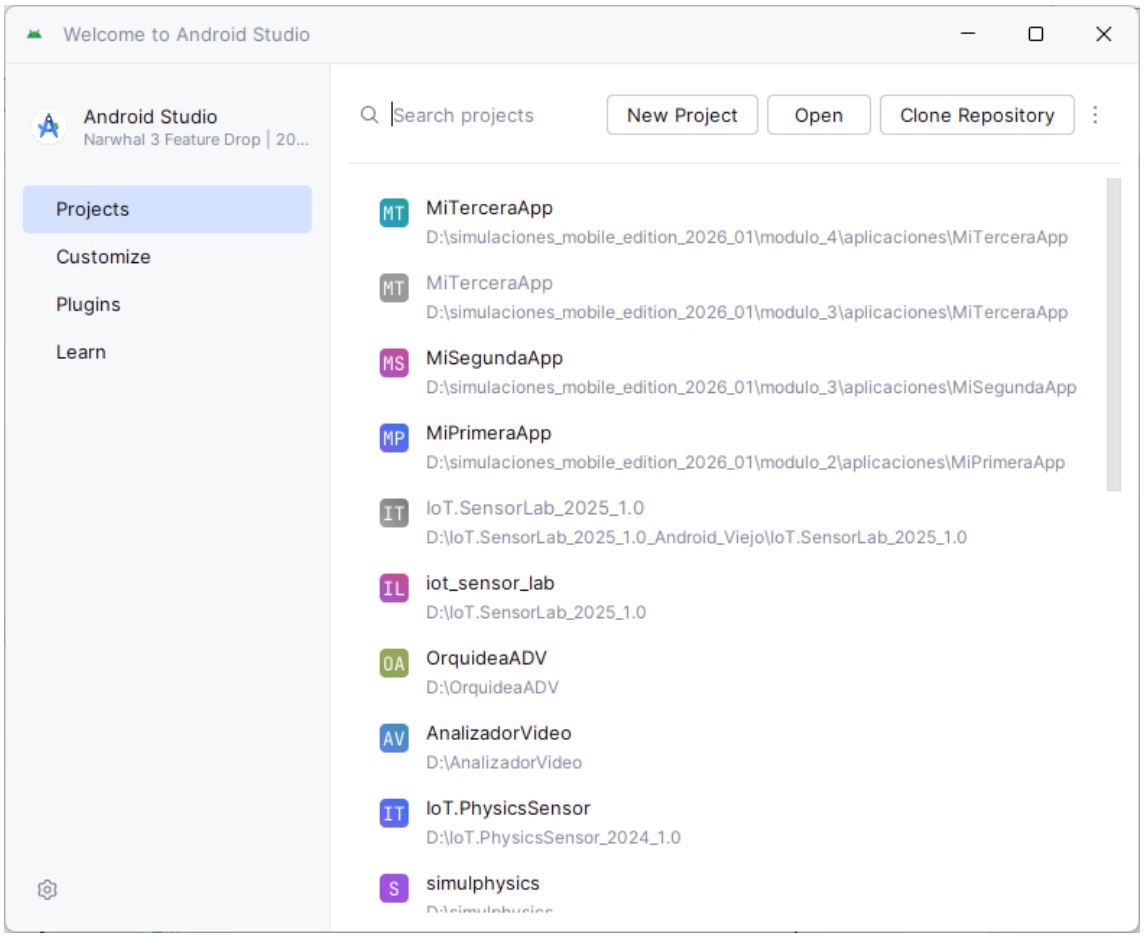

Figura 32

Paso 2:

Hacer clic en **New Project** para iniciar el proyecto. Se despliega la interfaz de la Figura 33.

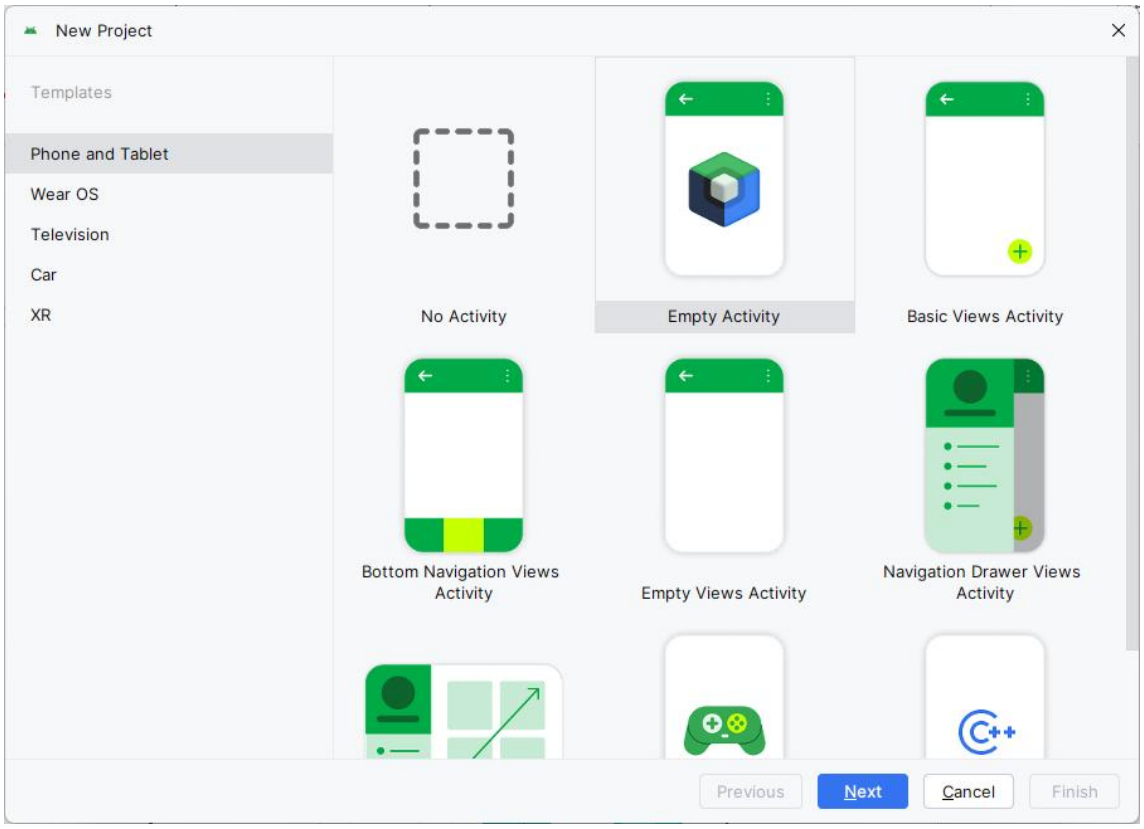

Figura 33

Paso 3:

Seleccionar **No Activity** y hacer clic en el botón **Next** lo cual desplegará la GUI de la Figura 34. Llenar los campos respectivos. El nombre de la aplicación debe comenzar con mayúsculas. En este caso se le denominó **MiCuartaApp**. Al dominio de la compañía se le denominó **com.curso_simulaciones**. Llenar el campo que elige el directorio donde se hospedará la aplicación. Luego el campo del lenguaje, en nuestro caso **Java**. Luego elegir la mínima Api que soportará la app, en nuestro caso elegimos la API 24 que nos garantiza que se nuestra app se ejecutará aproximadamente en el 98.6 % de los dispositivos ANDROID actuales. Hacer clic en **Finish** y se despliega la interfaz de la Figura 35.

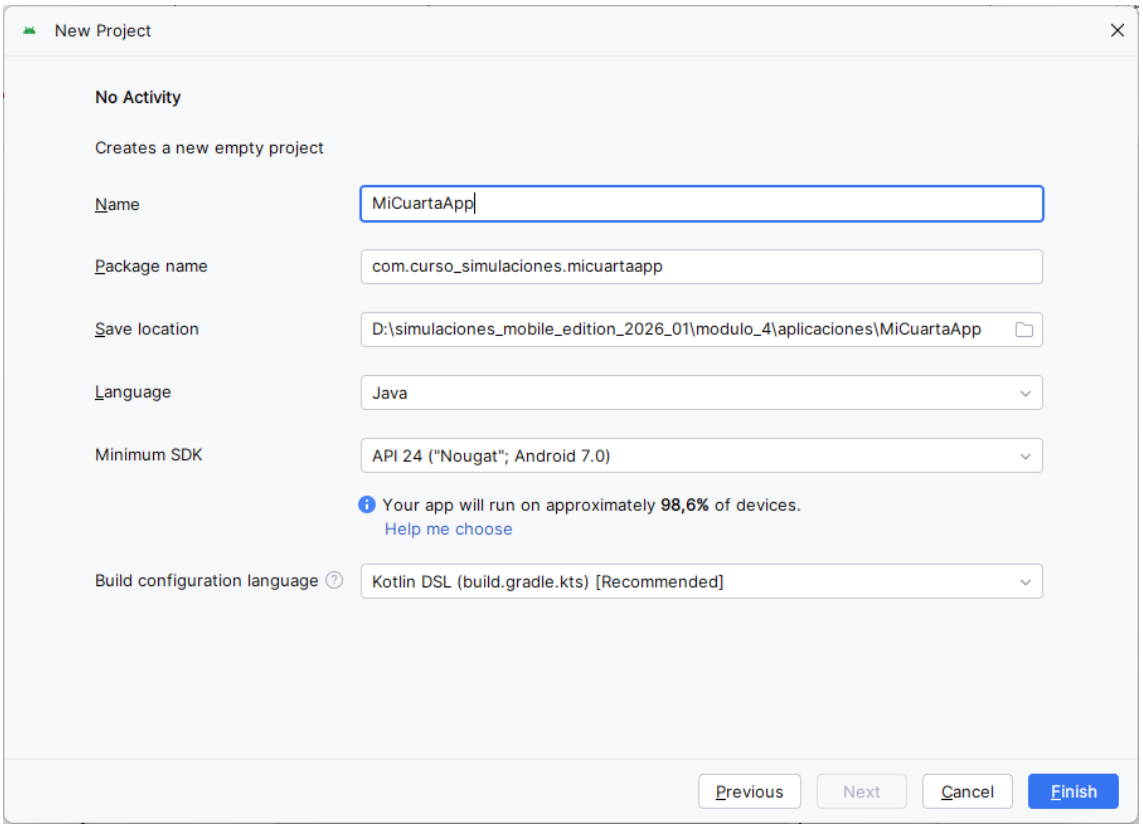

Figura 34

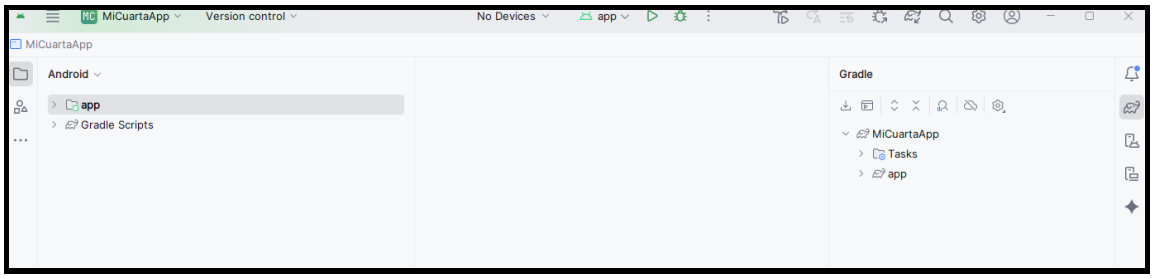

Figura 35

Paso 4: Creación de la clase ActividadPrincipalMiCuartaApp

Ubicarse sobre **java/com/curso_simulaciones>micuartaapp**, y hacer clic derecho. Se despliega la interfaz de la Figura 36,

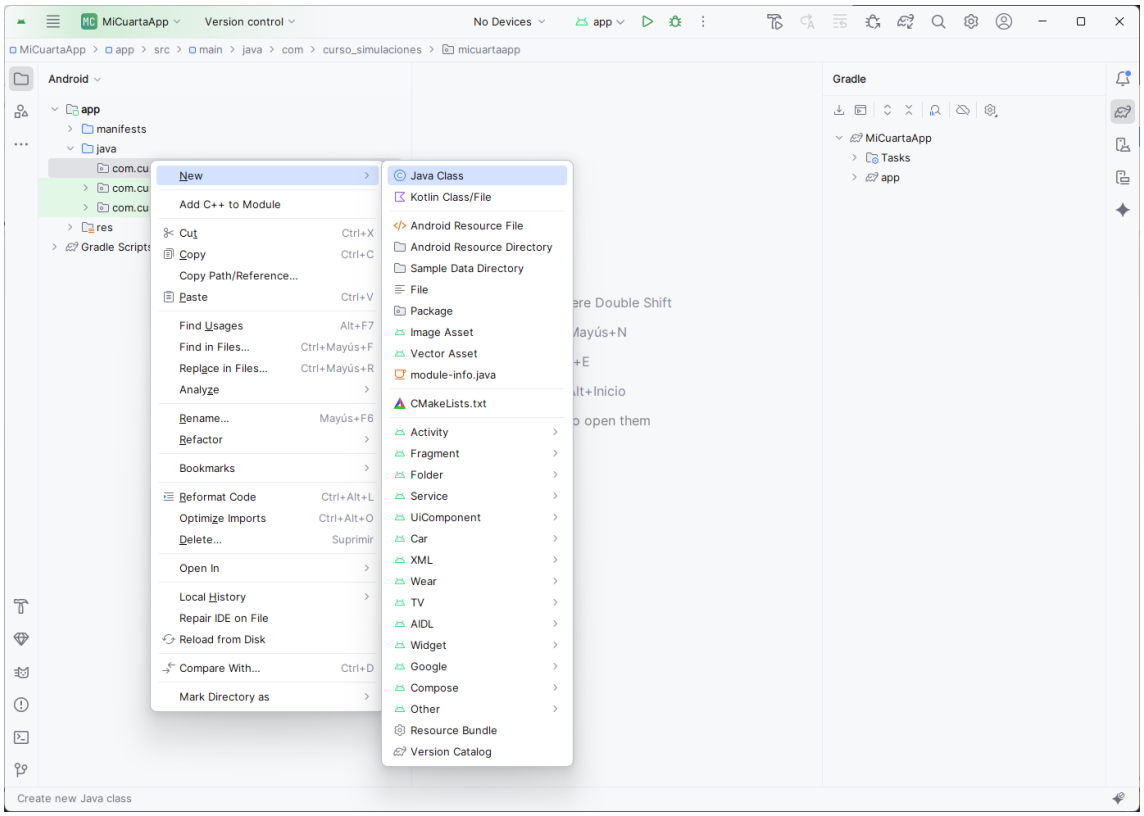

Figura 36

Seleccionar **New > Java Class**. Se despliega la interfaz de la Figura 37. En el campo escribir el nombre de la clase que se creará, en este caso se eligió el nombre **ActividadPrincipalMiCuartaApp**. Hacer doble clic **Class**. Se despliega la interfaz de la Figura 38.

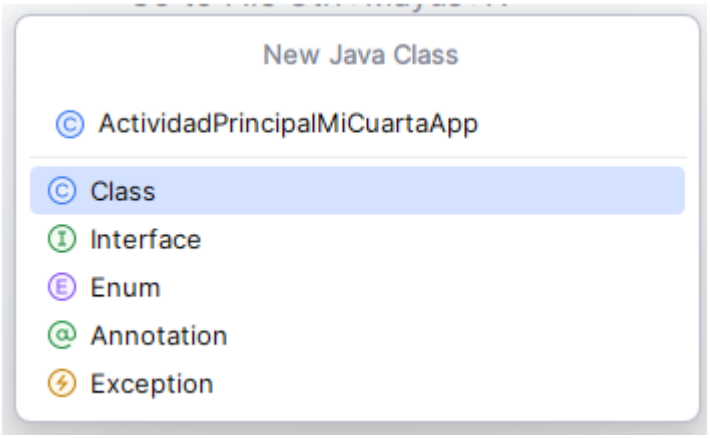

Figura 37

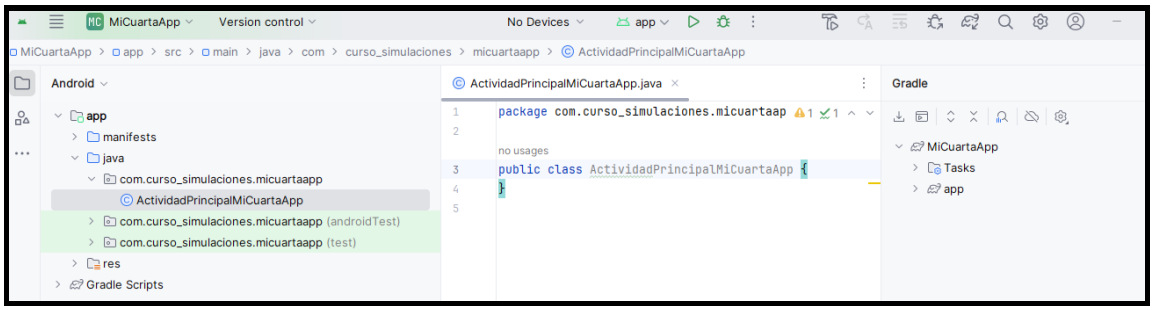

Figura 38

Paso 5: Creación del paquete componentes

Ubicarse sobre **java/com/curso_simulaciones>micuartaapp**, hacer clic derecho y **New > Package**, Figura 39. Se despliega la interfaz de la Figura 40. En el campo agregar el nombre del paquete, **componentes**. Hacer **ENTER**. Se despliega la interfaz de la Figura 41.

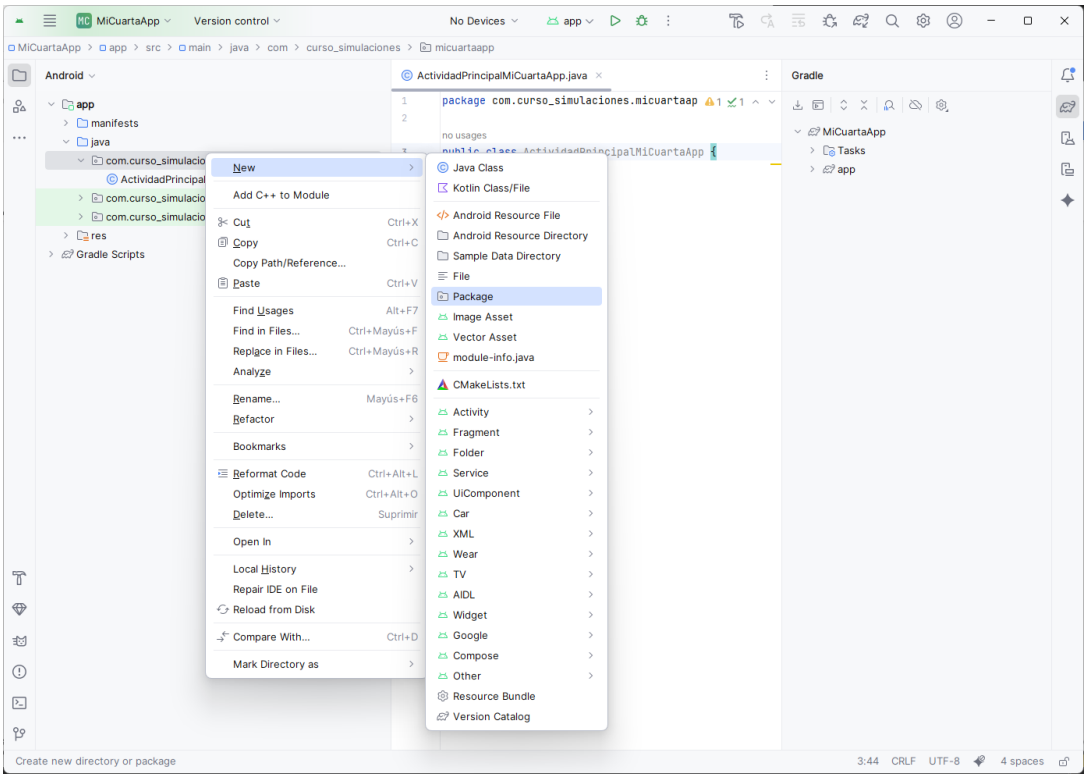

Figura 39

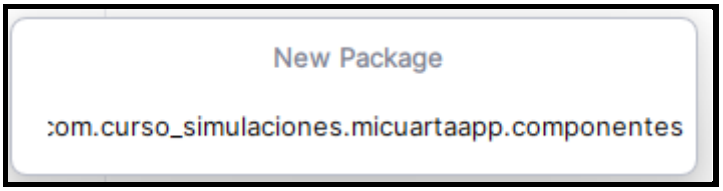

Figura 40

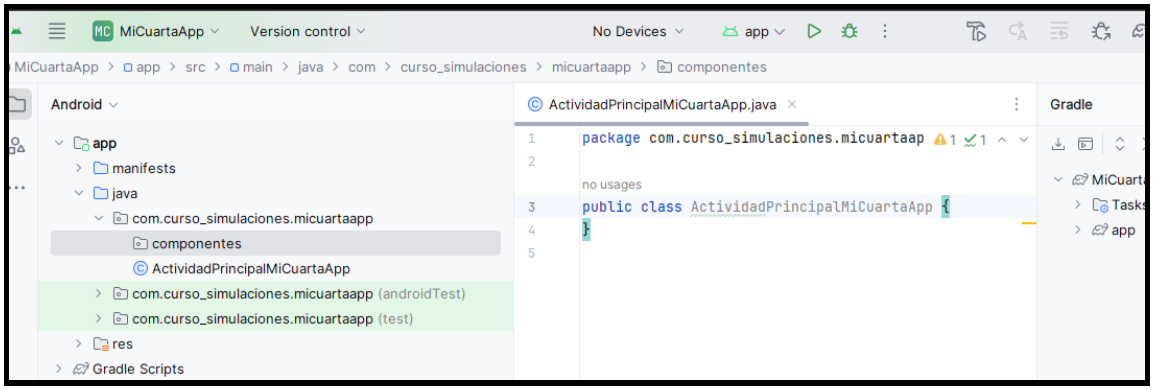

Figura 41

Paso 6: Creación de la clase GaugeSimple

Ubicar el cursor en **componentes** hacer clic derecho, seleccionar **New > Java Class** y en el campo escribir el nombre de la clase, **GaugeSimple**. Hacer clic en **Class**. La GUI se transforma en la de la Figura 42.

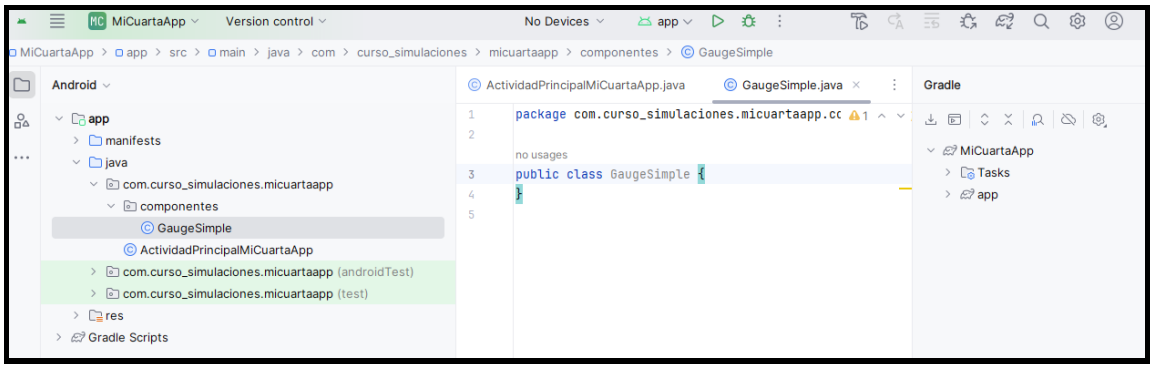

Figura 42

Paso 7: Agregar el código a la clase GaugeSimple

A la clase **GaugeSimple** agregar el siguiente código.

```
public class GaugeSimple extends View {

    private float largo;
    private float minimo = 0;
    private float maximo = 100f;
    private float medida = 40.0f;//tomar como medida inicial
    private String unidades = "UNIDADES";
    private int colorPrimerTercio = Color.rgb(200, 200, 0);
    private int colorSegundoTercio = Color.rgb(0, 180, 0);
    private int colorTercerTercio = Color.RED;
    private int colorLineas = Color.BLACK;
```

```
    private int colorFondo = Color.WHITE;
    private int colorNumerosDesplieggue = Color.BLACK;
    private int colorFranjaDinamica = Color.rgb(0, 0, 255);
    private int angPrimertercio = 100;
    private int angSegundoTercio = 100;
    private int angTercerTercio = 50;


    /**
     * Constructor de GaugeSimple
     */
    public GaugeSimple(Context context) {

        super(context);

        if (android.os.Build.VERSION.SDK_INT >=
android.os.Build.VERSION_CODES.HONEYCOMB) {
            this.setLayerType(View.LAYER_TYPE_SOFTWARE, null);

        }
    }

    /**
     * Modifica el rango de medicion
     * desde minimo hasta maximo
     *
     * @param minimo
     * @param maximo
     */
    public void setRango(float minimo, float maximo) {

        this.minimo = minimo;
        this.maximo = maximo;

    }


    /**
     * Modifica el valor medido
     *
     * @param medida
     */
    public void setMedida(float medida) {

        this.medida = medida;

    }


    /**
     * Regresa el valor medido
     *
     * @return medida
     */
    public float getMedida() {

        return medida;
    }


    /**
     * Modifica las unidades del instrumento virtual
     *
     * @param unidades
     */
    public void setUnidades(String unidades) {

        this.unidades = unidades;

    }


    /**
     * Modifica los colores de los sectores circulares
     *
     * @param colorPrimerTercio
```

```
     * @param colorSegundoTercio
     * @param colorTercerTercio
     */
    public void setColorSectores(int colorPrimerTercio, int colorSegundoTercio, int
colorTercerTercio) {

        this.colorPrimerTercio = colorPrimerTercio;
        this.colorSegundoTercio = colorSegundoTercio;
        this.colorTercerTercio = colorTercerTercio;

    }

    /**
     * Modifica los angulos de los sectores circulares
     * Deben sumar 250 grados
     *
     * @param angPrimerTercio
     * @param angSegundoTercio
     * @param angTercerTercio
     */
    public void setAngulosSectores(int angPrimerTercio, int angSegundoTercio, int
angTercerTercio) {
        this.angPrimertercio = angPrimerTercio;
        this.angSegundoTercio = angSegundoTercio;
        this.angTercerTercio = angTercerTercio;

    }

    public void setColorFranjaDinámica(int colorFranjaDinamica) {

        this.colorFranjaDinamica = colorFranjaDinamica;


    }

    /**
     * Modifica el color de fondo del tacometro
     *
     * @param colorFondo
     */
    public void setColorFondoTacometro(int colorFondo) {

        this.colorFondo = colorFondo;


    }


    /**
     * Modifica el color de las lineas del tacometro
     *
     * @param color_lineas
     */
    public void setColorLineasTacometro(int color lineas) {

        this.colorLineas = color_lineas;


    }


    /**
     * Modifica el color del numero que se despliega
     *
     * @param colorNumerosDesplieggue
     */

    public void setColorNumeroDespliegue(int colorNumerosDesplieggue) {

        this.colorNumerosDesplieggue = colorNumerosDesplieggue;

    }


    /**
     * @param canvas
```

```
     */

    //método para dibujar
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);

        /*
        se graba el estado actual del canvas
        para al final restaurarlo
        */
        canvas.save();


         /*
         La vista tendra las mismas dimensiones de su
         contenedor
         */
        float ancho = this.getWidth();//ancho de la vista
        float alto = this.getHeight();//alto de la vista

        /*
         Se define la variable largo como el 80%
         del menor valor entre alto y largo del
         contenedor
         */

        if (ancho > alto) {

            largo = 0.8f * alto;

        } else {

            largo = 0.8f * ancho;


        }

        /*
          se hace tralación del (0,0) al centro
          del contenedor
        */
        canvas.translate(0.5f * ancho, 0.5f * alto);

        //configurando el pincel
        Paint pincel = new Paint();
        //evita efecto sierra
        pincel.setAntiAlias(true);
        //tamaño texto
        pincel.setTextSize(0.05f * largo);
        //para mejor manejo de la métrica de texto
        pincel.setLinearText(true);
        //para efectos de buen escalado de bitmaps
        pincel.setFilterBitmap(true);
        //para buen manejo de gradientes de color
        pincel.setDither(true);


        float esquinaSuperiorIzquierdaX = -0.5f * largo;
        float esquinaSuperiorIzquierdaY = -0.5f * largo;
        float esquinaInferiorDerechaX = 0.5f * largo;
        float esquinaInferiorDerechaY = 0.5f * largo;

        //dibujar los tres segementos circulares
        RectF rect = new RectF(esquinaSuperiorIzquierdaX, esquinaSuperiorIzquierdaY,
                esquinaInferiorDerechaX, esquinaInferiorDerechaY);

        pincel.setColor(colorPrimerTercio);
        canvas.drawArc(rect, 145, angPrimertercio, true, pincel);
        pincel.setColor(colorSegundoTercio);
        canvas.drawArc(rect, 145 + angPrimertercio, angSegundoTercio, true, pincel);
        pincel.setColor(colorTercerTercio);
        canvas.drawArc(rect, 145 + angPrimertercio + angSegundoTercio, angTercerTercio,
true, pincel);


        float indent = (float) (0.05 * largo);
```

```
        float posicionY = (float) (0.5 * largo);

        //franja dinámica
        //calcular angulo para ubicar la aguja de acuerdo al valor medido
        float angulo_rotacion_medida = 235 + (250f / (maximo - minimo)) * (medida -
minimo);
        float a = (float) 0.01 * largo;
        rect = new RectF(esquinaSuperiorIzquierdaX + a, esquinaSuperiorIzquierdaY + a,
                esquinaInferiorDerechaX - a, esquinaInferiorDerechaY - a);
        pincel.setColor(colorFranjaDinamica);
        canvas.drawArc(rect, 145, angulo_rotacion_medida - 235, true, pincel);


        /*
         dibujar el tacometro sin la aguja
        */
        //aqui empieza el dibujo
        float radio = (float) (0.48 * largo);
        pincel.setColor(colorFondo);
        canvas.drawCircle(0, 0, radio, pincel);
        pincel.setColor(colorLineas);
        pincel.setStyle(Paint.Style.STROKE);
        canvas.drawCircle(0, 0, radio, pincel);
        pincel.setStrokeWidth(1f);
        canvas.drawCircle(0, 0, 0.5f * largo, pincel);

        //grosor de las líneas
        pincel.setStrokeWidth(0.01f * largo);



         /*
         Divisiones grandes
         Se dibuja primero la división vertical.
         Luego se repite rotando de a 50 grados comenzando
         en 235 grados.
          */
        for (int i = 0; i < 6; i = i + 1) {
            float anguloRotacion = 235 + 50 * i;
            canvas.rotate(anguloRotacion, 0, 0);
            canvas.drawLine(0, -posicionY, 0, -posicionY + indent, pincel);
            canvas.rotate(-anguloRotacion, 0, 0);

            /*
             Dibujar los números
            */
            int valorIncrementoMarcas = (int) ((maximo - minimo) / 5f);
            int valorMarca = (int) (minimo + valorIncrementoMarcas * i);
            String numero = "" + valorMarca;

            //ancho de la cadena del número
            float anchoCadenaNumero = pincel.measureText(numero);
            //alto de la cadena del número
            Rect frontera = new Rect();
            pincel.getTextBounds(numero, 0, numero.length(), frontera);
            int altoCadenaNumero = frontera.height();

            //cálculo de corrdenadas de la posicion (p_x,p_y)
            //del número segun la marcas de divisiones grandes
            double angulo = Math.toRadians(anguloRotacion);
            float radio_posicion_numero= 0.5f*largo-1.4f*indent;
            int p_x = (int)(radio_posicion_numero*Math.sin(angulo));
            int p_y = (int)(radio_posicion_numero*Math.cos(angulo)-
0.5f*altoCadenaNumero);
            float d_x =0;

            if (p_x<0 || p_x==0) {
                d_x = -0.2f*anchoCadenaNumero;
            } else {
                d_x= - 0.8f*anchoCadenaNumero;
            }


            //dibujar la letra rellenita
            pincel.setStyle(Paint.Style.FILL);
```

```
            //dibujar los números de marcas
            canvas.translate(p_x, -p_y);
            canvas.drawText(numero, d_x, 0,pincel);
            canvas.translate(-p x, p y);

        }

        /*
         Divisiones pequeñas
         Se dibuja primero la división vertical.
         Luego se repite rotando de a 10 grados comenzando
         en 235 grados.
         */
        pincel.setStyle(Paint.Style.STROKE);
        //grosor de las líneas
        pincel.setStrokeWidth(0.005f * largo);
        for (int i = 0; i < 26; i = i + 1) {

            float anguloRotacion = 235 + 10 * i;
            canvas.rotate(anguloRotacion, 0, 0);
            canvas.drawLine(0, -posicionY, 0, -posicionY + (float) (0.6 * indent),
pincel);
            canvas.rotate(-anguloRotacion, 0, 0);

        }
        //aqui termina dibujo del tacometro sin aguja



        /*
        dibujar la aguja
        */
        //aqui empieza dibujo de la aguja
        pincel.setStrokeWidth(0.005f * largo);
        pincel.setColor(Color.RED);
        canvas.rotate(angulo_rotacion_medida, 0, 0);
        float b = (float) (1.5f* indent);
        canvas.drawLine(0, -posicionY, 0, b, pincel);
        canvas.rotate(-angulo_rotacion_medida, 0, 0);
        pincel.setStyle(Paint.Style.FILL);
        pincel.setColor(colorFondo);
        canvas.drawCircle(0, 0, (float) (0.4 * indent), pincel);
        pincel.setColor(Color.RED);
        pincel.setStyle(Paint.Style.STROKE);
        canvas.drawCircle(0, 0, (float) (0.4 * indent), pincel);

        //dibujar circulo apoyo aguja

        //aquí termina dibujo de la aguja

        //Dibujar las unidades
        pincel.setStyle(Paint.Style.FILL);
        pincel.setColor(colorLineas);
        pincel.setTextSize(0.08f * largo);
        float anchoCadenaUnidades = pincel.measureText(unidades);
        canvas.drawText(unidades, -0.5f * anchoCadenaUnidades,- 0.15f * largo, pincel);
        //aqui termina dibujo de las unidades

        //aqui despliegue de la medida
        pincel.setTextSize(0.1f * largo);
        float anchoCadenaNumero = pincel.measureText("" + medida);
        pincel.setColor(colorNumerosDesplieggue);
        canvas.drawText("" + medida, -0.5f * anchoCadenaNumero, 0.2f * largo, pincel);

        //marcar empresa
        String empresa = "IoT.PhysicsSensor";
        pincel.setTextSize(0.05f * largo);
        float anchoCadenaNombreEmpresa = pincel.measureText(empresa);
        canvas.drawText(empresa, -0.5f * anchoCadenaNombreEmpresa, 0.35f * largo,
pincel);



        //se restaura el canvas al estado incial
        //el que se garbó al principio de este método
        canvas.restore();
```

```
        //para efectos de animación
        invalidate();

    }//fin onDraw
}
```

Paso 8:

Agregar el código a la clase **ActividadPrincipalMiCuartaApp**


```
public class ActividadPrincipalMiCuartaApp extends Activity {

    private GaugeSimple tacometro_1, tacometro_2,tacometro_3;


    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

    /*llamada al método para crear los elementos de la interfaz
    gráfica de usuario (GUI)*/
        crearElementosGui();


    /*para informar cómo se debe adaptar la GUI a la pantalla del
    dispositivo*/
        ViewGroup.LayoutParams parametro_layout_principal = new
ViewGroup.LayoutParams(ViewGroup.LayoutParams.MATCH_PARENT,
ViewGroup.LayoutParams.MATCH_PARENT);


    /*pegar al contenedor la GUI: en el argumento se está llamando
    al método crearGui()*/
        this.setContentView(crearGui(), parametro_layout_principal);

    }


    /*crear los objetos de la interfaz gráfica de usuario (GUI)*/
    private void crearElementosGui() {

        //crear objeto GaugeSimple
        tacometro_1=new GaugeSimple(this);
        //cambiar atributos (propiedades)
        //darle color blanco al lienzo antes de pega
        tacometro_1.setBackgroundColor(Color.WHITE);
        //asignar las unidades
        tacometro_1.setUnidades("HR %");
        //asignar rangos
```

```
        tacometro_1.setRango(0,100);
        //asignar la medida
        tacometro_1.setMedida(78f);

        //crear objeto GaugeSimple
        tacometro_2=new GaugeSimple(this);
        //cambiar atributos (propiedades)
        //darle color blanco al lienzo antes de pegar
        tacometro_2.setBackgroundColor(Color.WHITE);
        //darle color a los sectores
        tacometro_2.setColorSectores(Color.RED, Color.BLUE, Color.rgb(200,200,20));
        //asignar las unidades
        tacometro_2.setUnidades("lx");
        //asignar rangos
        tacometro_2.setRango(0,1000);
        //asignar la medida
        tacometro_2.setMedida(120f);

        //crear objeto GaugeSimple
        tacometro_3=new GaugeSimple(this);
        //cambiar atributos (propiedades)
        //darle color blanco al lienzo antes de pegar
        tacometro_3.setBackgroundColor(Color.WHITE);
        //darle color a los sectores
        tacometro_3.setColorSectores(Color.GREEN, Color.GREEN, Color.rgb(255,100,0));
        //asignar las unidades
        tacometro_3.setUnidades("ax en m/s.s");
        //asignar rangos
        tacometro_3.setRango(0,10);
        //asignar la medida
        tacometro_3.setMedida(6.5f);

    }


    /*organizar la distribución de los objetos de de la GUI usando
    administradores de diseño*/
    private LinearLayout crearGui() {

        //administrador de diseño
        LinearLayout linear_principal = new LinearLayout(this);
        linear_principal.setOrientation(LinearLayout.HORIZONTAL);
        linear_principal.setGravity(Gravity.CENTER_HORIZONTAL);
        linear_principal.setGravity(Gravity.FILL);
        linear_principal.setBackgroundColor(Color.rgb(250, 150, 50));
        linear_principal.setWeightSum(3);

        LinearLayout linear_izquierdo = new LinearLayout(this);
        linear_izquierdo.setOrientation(LinearLayout.VERTICAL);
        linear_izquierdo.setGravity(Gravity.CENTER_HORIZONTAL);
        linear_izquierdo.setGravity(Gravity.FILL);
```

```
        linear_izquierdo.setBackgroundColor(Color.RED);
        linear_izquierdo.setWeightSum(1);

        LinearLayout linear_centro = new LinearLayout(this);
        linear_centro.setOrientation(LinearLayout.VERTICAL);
        linear_centro.setGravity(Gravity.CENTER_HORIZONTAL);
        linear_centro.setGravity(Gravity.FILL);
        linear_centro.setBackgroundColor(Color.BLUE);
        linear_centro.setWeightSum(1);

        LinearLayout linear_derecho = new LinearLayout(this);
        linear_derecho.setOrientation(LinearLayout.VERTICAL);
        linear_derecho.setGravity(Gravity.CENTER_HORIZONTAL);
        linear_derecho.setGravity(Gravity.FILL);
        linear_derecho.setBackgroundColor(Color.GREEN);
        linear_derecho.setWeightSum(1);


        //parametro para pegar los gauges
        LinearLayout.LayoutParams parametrosPegadaGauges= new
LinearLayout.LayoutParams(android.view.ViewGroup.LayoutParams.MATCH_PARENT,0);
        parametrosPegadaGauges.setMargins(20, 20, 20, 20);
        parametrosPegadaGauges.weight = 1.0f;

        //pegar gauges
        linear_izquierdo.addView(tacometro_1,parametrosPegadaGauges);
        linear_centro.addView(tacometro_2,parametrosPegadaGauges);
        linear_derecho.addView(tacometro_3,parametrosPegadaGauges);


        //parametro para pegar los linear al pricipal
        LinearLayout.LayoutParams parametrosPegadaLinear= new
LinearLayout.LayoutParams(0,android.view.ViewGroup.LayoutParams.MATCH_PARENT);
        parametrosPegadaLinear.setMargins(20, 20, 20, 20);
        parametrosPegadaLinear.weight = 1.0f;

        linear_principal.addView(linear_izquierdo,parametrosPegadaLinear);
        linear_principal.addView(linear_centro,parametrosPegadaLinear);
        linear_principal.addView(linear_derecho,parametrosPegadaLinear);

        return linear_principal;

    }

}
```

Paso 9: Agregar el código al archivo manifiesto

Hay que recordar que toda Activity se debe declarar en el archivo manifiesto. Ir al **AndroidManifest.xml**.

Ahora es necesario agregar código en el archivo manifiesto. Hacer clic en el archivo **app.manifests> AndroidManifest.xml**. En el código desplegado agregar código de tal forma que el archivo quede como el siguiente.


```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/Theme.MiCuartaApp"
        tools:targetApi="31" >

    <activity
android:name="com.curso_simulaciones.micuartaapp.ActividadPrincipalMiCuartaApp"
        android:exported="true"
        android:screenOrientation="landscape">

        <intent-filter>
            <action android:name="android.intent.action.MAIN" />

            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>

    </application>

</manifest>
```

Ya se está en condiciones de proceder a **compilar** y **ejecutar** la aplicación.

Paso 10

Conectar el dispositivo por un puerto USB del PC. Proceder a la **compilación y ejecución** haciendo clic en el botón de la barra superior de herramientas de Android Studio que está señalado con un círculo rojo en la Figura 43. Se despliega la GUI de la Figura 44.

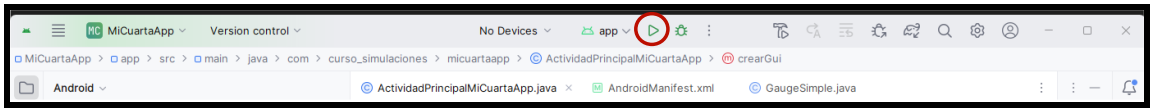

Figura 43

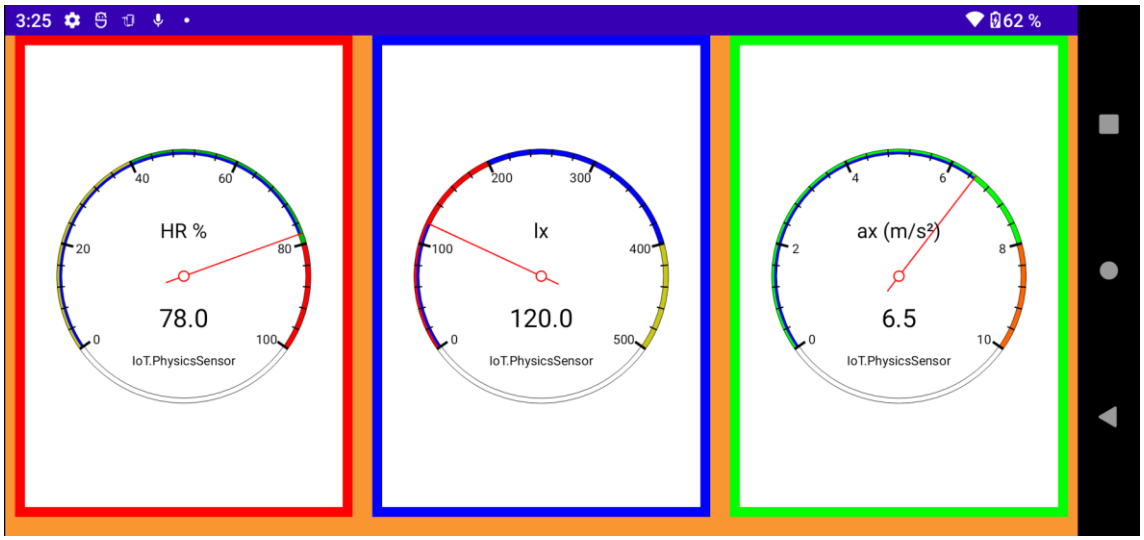

Figura 44

DISTRIBUYENDO LA APP

Se ejecuta **Build APK (S)**, Figura 45.

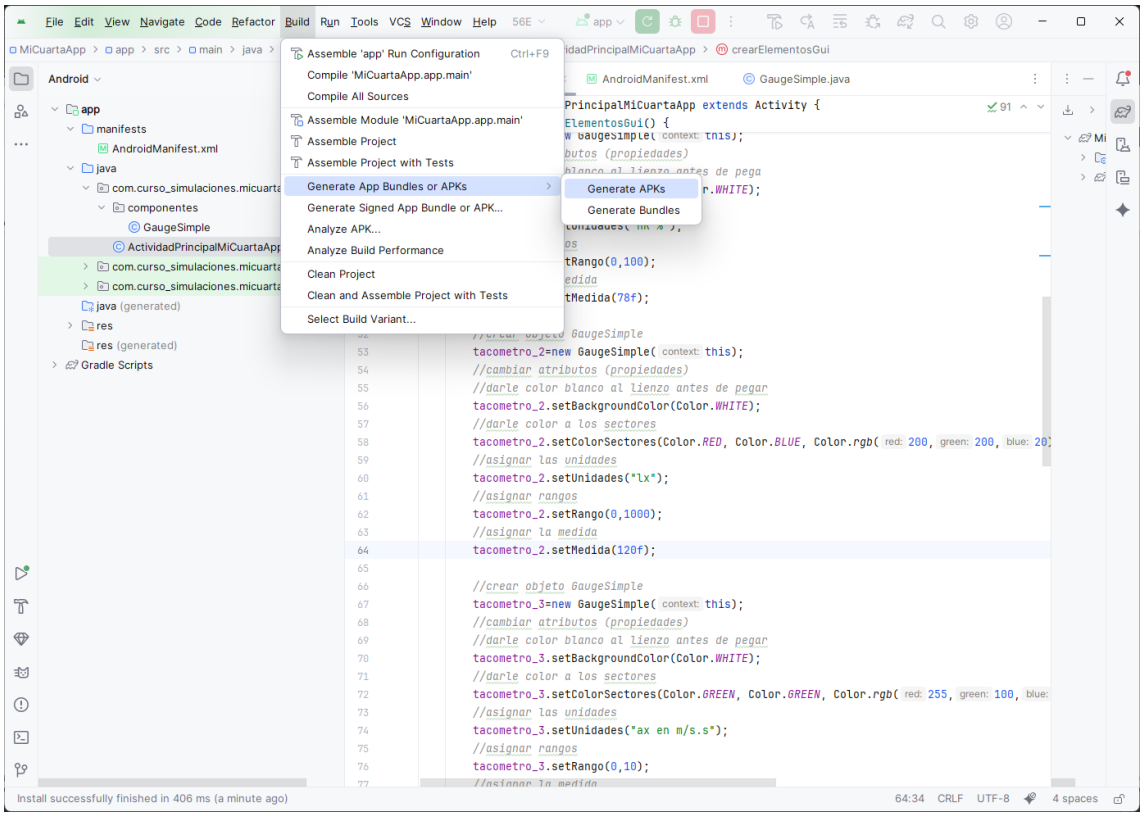

Figura 45

El archivo del programa para distribuir se encuentra en la carpeta donde se hospeda la aplicación (tiene extensión **.apk**):

App > build > outputs > apk > debug > app-debug.apk

Si se desea se puede cambiar el nombre (es lo recomendable).

TAREA

Desarrollar una aplicación denominada **MiQuintaApp** que cumpla los mismos requisitos de **MiCuartaApp** pero donde el **GaugeSimple** tenga el aspecto de la Figura 46 (debe ser RESPONSIVO). Este debe tener métodos para cambiar los colores del fondo, de la aguja, de los números, de la escala y de la unidades. También métodos para cambiar las unidades y la escala (mínimo y máximo). La app debe desplegar **tres instancias** de este gauge con difrentes colores, unidades y marcar valores diferentes.

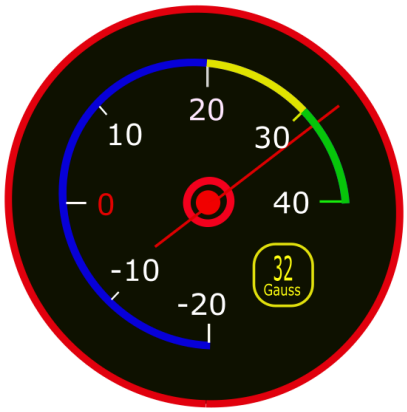

Figura 46