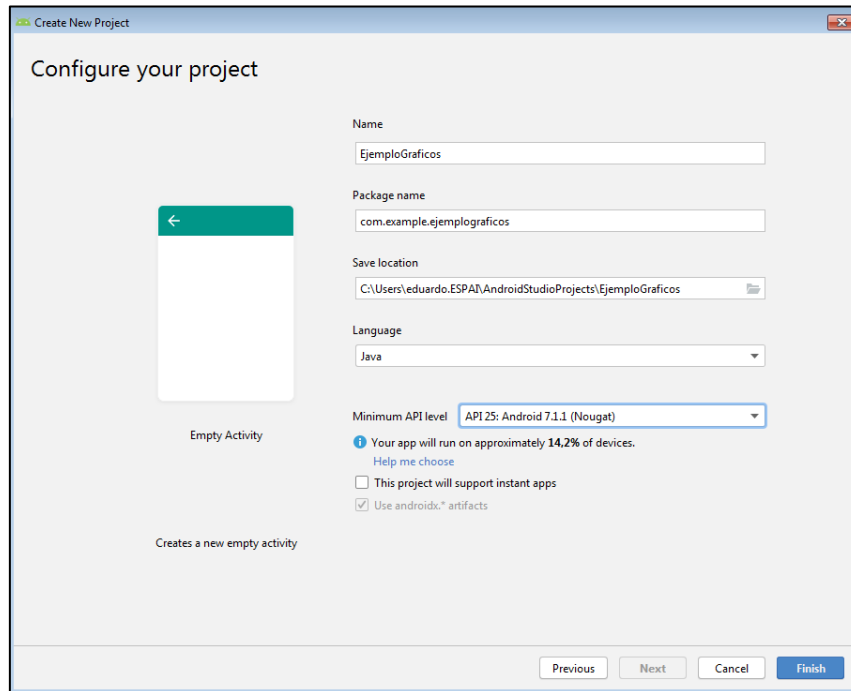


## PRACTICA 1: EJEMPLOGRAFICOS

### Parte I: Creación de una vista personalizada

A continuación se muestra un ejemplo donde se crea una vista que es dibujada por código por medio de un Canvas.

**Paso 1.** Crea un nuevo proyecto, de nombre EjemploGraficos, con los siguientes datos:



**Paso 2.** Reemplaza el código de la actividad por:

```
EjemploGraficosActivity.java x
package com.example.ejemplograficos;

import ...

public class EjemploGraficosActivity extends AppCompatActivity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(new EjemploView( context: this));
    }

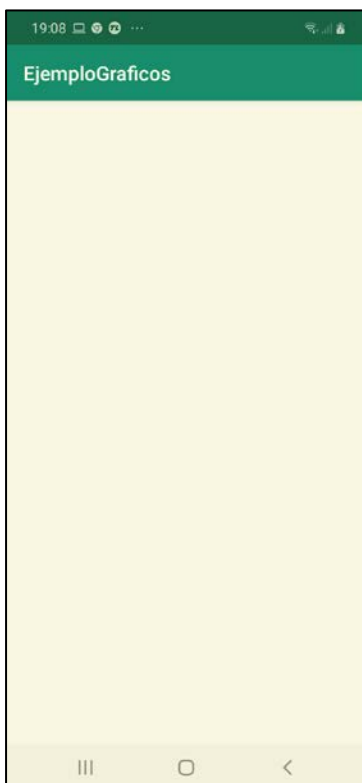
    public class EjemploView extends View {
        public EjemploView (Context context) {
            super(context);
        }
        @Override
        protected void onDraw(Canvas canvas) {
            //Dibujar aquí
        }
    }
}
```

```
AndroidManifest.xml
1 <?xml version="1.0" encoding="utf-8"?>
2 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3     package="com.example.ejemplograficos">
4
5     <application
6         android:allowBackup="true"
7         android:icon="@mipmap/ic_launcher"
8         android:label="EjemploGraficos"
9         android:roundIcon="@mipmap/ic_launcher_round"
10        android:supportsRtl="true"
11        android:theme="@style/AppTheme">
12        <activity android:name=".EjemploGraficosActivity">
13            <intent-filter>
14                <action android:name="android.intent.action.MAIN" />
15
16                <category android:name="android.intent.category.LAUNCHER" />
17            </intent-filter>
18        </activity>
19    </application>
20
21 </manifest>
```

Comienza con la creación de una Activity, pero en este caso, el objeto View que asociamos a la actividad mediante el método setContentView() no está definido mediante XML. Por el contrario, es creado mediante código a partir del constructor de la clase EjemploView.

La clase EjemploView extiende View, modificando solo el método onDraw() responsable de dibujar la vista. A lo largo del capítulo iremos viendo ejemplos de código que pueden ser escritos en este método.

**Paso 3.** Si ejecutas la aplicación no se observará nada. Aprenderemos a dibujar en esta Canvas utilizando el objeto Paint descrito en la siguiente sección.



**Nota sobre Java:** Siempre que en un ejemplo aparezca un error indicándote que no se puede resolver un tipo, pulsa **Alt+Intro** para añadir los import de forma automática.

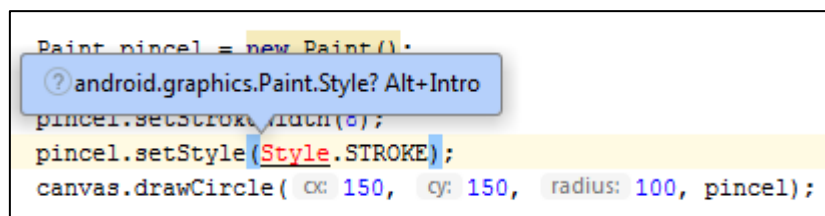
## Parte II: Uso de la clase Paint

**Paso 1.** Escribe dentro de OnDraw del ejercicio anterior el código siguiente:



```
1 EjemploGraficosActivity.java x
2
3 public class EjemploGraficosActivity extends AppCompatActivity {
4
5     @Override
6     public void onCreate(Bundle savedInstanceState) {
7         super.onCreate(savedInstanceState);
8         setContentView(new EjemploView( context: this));
9     }
10
11     public class EjemploView extends View {
12         public EjemploView (Context context) { super(context); }
13         @Override
14         protected void onDraw(Canvas canvas) {
15
16             Paint pincel = new Paint();
17             pincel.setColor(Color.BLUE);
18             pincel.setStrokeWidth(8);
19             pincel.setStyle(Paint.Style.STROKE);
20             canvas.drawCircle( cx: 150, cy: 150, radius: 100, pincel);
21         }
22     }
23 }
```

**Nota sobre Java:** Si pulsas **Alt+Intro** para añadir los import el sistema te advertirá que la clase Style está definida en dos paquetes diferentes:

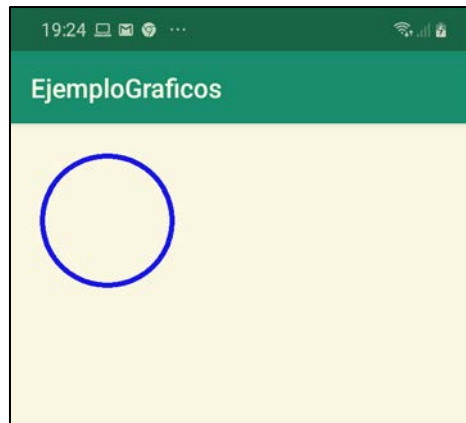


```
Paint pincel = new Paint();
pincel.setStrokeWidth(8);
pincel.setStyle(Style.STROKE);
canvas.drawCircle( cx: 150, cy: 150, radius: 100, pincel);
```

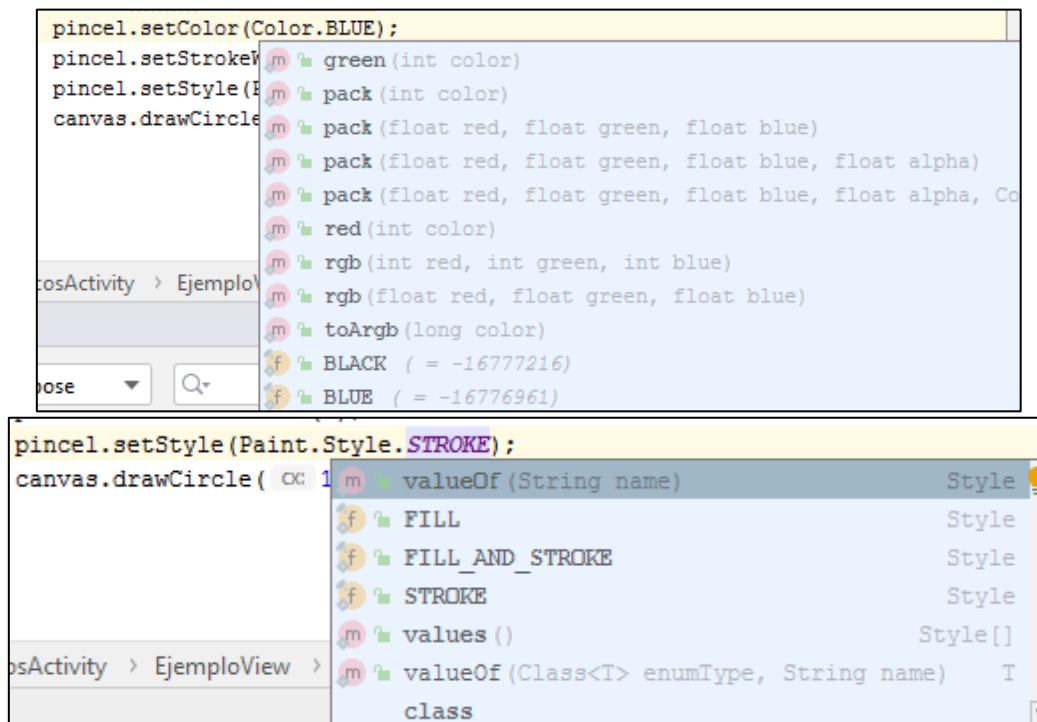
Te preguntará cual de los dos quieres importar. No suele resultar complicado resolver este problema si analizas el contexto en el que estás trabajando. En nuestro caso estamos utilizando la clase Paint por lo que la primera opción es la adecuada. Si alguna vez te equivocas en esta selección, lo normal es que aparecerán errores en el código. En este caso, deshaces y seleccionas la otra opción.

```
import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.os.Bundle;
import android.view.View;
```

**Paso 2.** Ejecuta la aplicación para ver el resultado.



**Paso 3.** Aprovechando la opción de autocompletar de Android Studio, prueba otros valores para Color y Style.



**Paso 4.** Prueba otros métodos de dibujo, como `drawLine()` o `drawPoint()`.



```
@Override
protected void onDraw(Canvas canvas) {

    Paint pincel = new Paint();
    pincel.setColor(Color.BLUE);
    pincel.setStrokeWidth(8);
    pincel.setStyle(Paint.Style.STROKE);
    canvas.drawCircle( 150, 150, 100, pincel);
    pincel.setColor(Color.MAGENTA);
    pincel.setStyle(Paint.Style.FILL_AND_STROKE);
    canvas.drawLine( 200, 200, 300, 300, pincel);
    pincel.setStrokeWidth(30);
    pincel.setColor(Color.YELLOW);
    canvas.drawPoint( 350, 350, pincel);
}
```

### Parte III: Definición de colores

**Paso 1.** Modifica el ejercicio anterior, añadiendo al final de OnDraw el código siguiente:

```
@Override
protected void onDraw(Canvas canvas) {

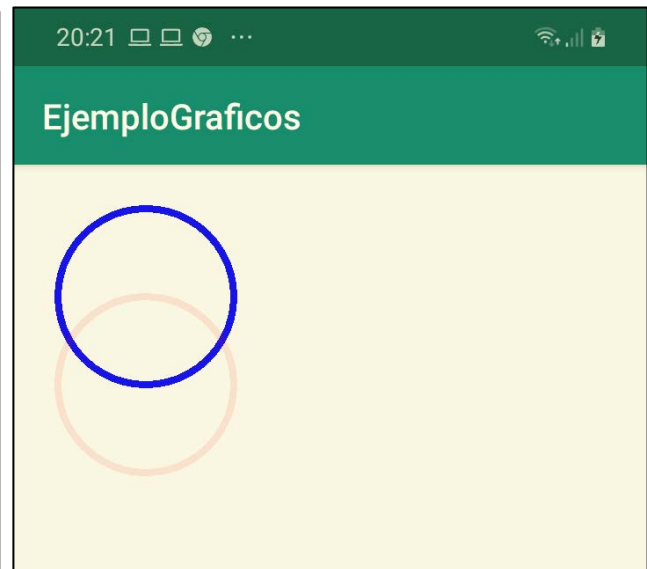
    Paint pincel = new Paint();
    pincel.setColor(Color.BLUE);
    pincel.setStrokeWidth(8);
    pincel.setStyle(Paint.Style.STROKE);
    canvas.drawCircle( cx: 150, cy: 150, radius: 100, pincel);

    pincel.setColor(Color.argb( alpha: 127, red: 255, green: 0, blue: 0));
    canvas.drawCircle( cx: 150, cy: 250, radius: 100, pincel);
}
```

**Paso 2.** Observa como el color rojo seleccionado es mezclado con el color de fondo. Prueba otros valores de alfa.

```
pincel.setColor(Color.argb(127,255,0,0));
canvas.drawCircle(150, 250, 100, pincel);
```

```
pincel.setColor(Color.argb(25,255,0,0));
canvas.drawCircle(150, 250, 100, pincel);
```



**Paso 3.** Reemplaza la primera línea que acabas de introducir por:

```
@Override
protected void onDraw(Canvas canvas) {

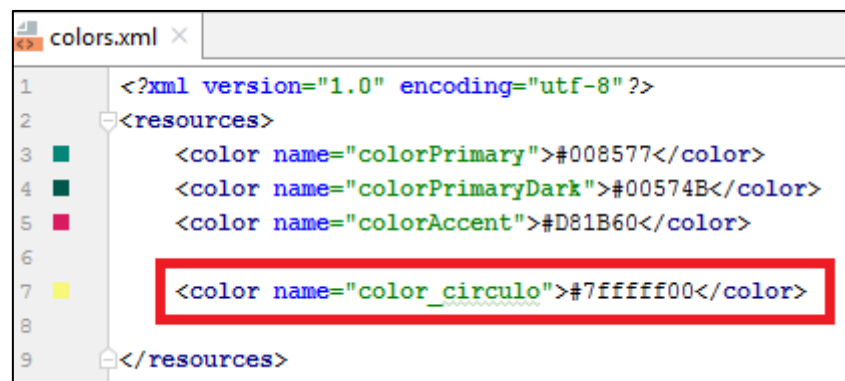
    Paint pincel = new Paint();
    pincel.setColor(Color.BLUE);
    pincel.setStrokeWidth(8);
    pincel.setStyle(Paint.Style.STROKE);
    canvas.drawCircle( cx: 150, cy: 150, radius: 100, pincel);

    //pincel.setColor(Color.argb(25,255,0,0));
    pincel.setColor(0x7FFF0000);
    canvas.drawCircle( cx: 150, cy: 250, radius: 100, pincel);
}
```

**Paso 4.** Observa como el resultado es idéntico.



**Paso 5.** Define en el fichero res/values/colors.xml un nuevo color utilizando el siguiente código:



**Paso 6.** Modifica el ejemplo anterior para que se utilice este color definido en los recursos:

```

//pincel.setColor(getResources().getColor(R.color.color_circulo, null));
pincel.setColor(EjemploGraficosActivity.this.getColor(R.color.color_circulo));
canvas.drawCircle(ox: 150, cy: 250, radius: 100, pincel);

```



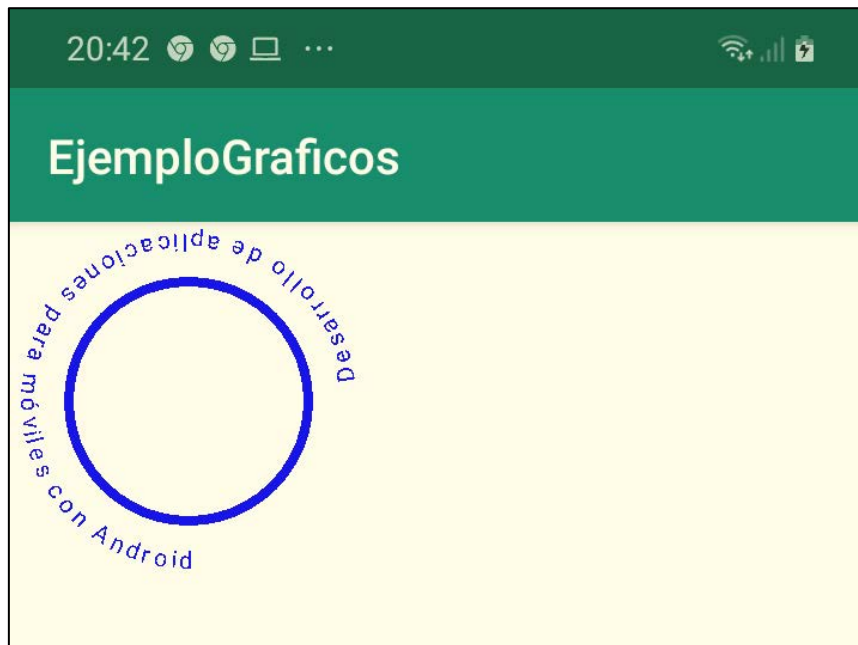
**Paso 7.** Haz una copia hasta aquí del onDraw(Canvas canvas), coméntalo añadiendo un comentario previo que anuncie que es el código de la partes 1 a 3. //PARTE 1, 2 y 3

## Parte IV: Uso de la clase Path

**Paso 1.** Reemplaza dentro de `OnDraw` el código siguiente:

```
public class EjemploView extends View {  
    public EjemploView (Context context) { super(context); }  
  
    @Override  
    protected void onDraw(Canvas canvas) {  
        Path trazo = new Path();  
        trazo.addCircle( x: 150, y: 150, radius: 100, Path.Direction.CCW);  
        canvas.drawColor(Color.WHITE);  
        Paint pincel = new Paint();  
        pincel.setColor(Color.BLUE);  
        pincel.setStrokeWidth(8);  
        pincel.setStyle(Style.STROKE);  
        canvas.drawPath(trazo, pincel);  
        pincel.setStrokeWidth(1);  
        pincel.setStyle(Style.FILL);  
        pincel.setTextSize(20);  
        pincel.setTypeface (Typeface.SANS_SERIF);  
        canvas.drawTextOnPath( text: "Desarrollo de aplicaciones para móviles con Android",  
                               trazo, hOffset: 10, vOffset: 40, pincel);  
    }  
}
```

**Paso 2.** El resultado obtenido ha de ser:

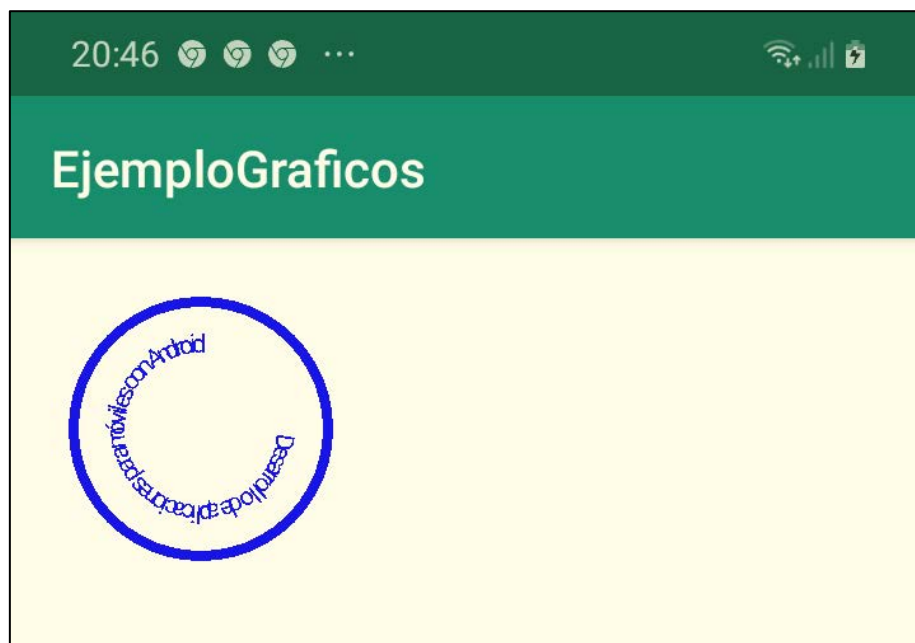


**Paso 3.** Modifica en la segunda línea el parámetro `Direction.CCW` (contrario a las agujas del reloj) por `Direction.CW` (a favor de las agujas del reloj). Observa el resultado.

```

@Override
protected void onDraw(Canvas canvas) {
    Path trazo = new Path();
    trazo.addCircle( x: 150, y: 150, radius: 100, Path.Direction.CW);
    canvas.drawColor(Color.WHITE);
    Paint pincel = new Paint();
    pincel.setColor(Color.BLUE);
    pincel.setStrokeWidth(8);
    pincel.setStyle(Style.STROKE);
    canvas.drawPath(trazo, pincel);
    pincel.setStrokeWidth(1);
    pincel.setStyle(Style.FILL);
    pincel.setTextSize(20);
    pincel.setTypeface(Typeface.SANS_SERIF);
    canvas.drawTextOnPath( text: "Desarrollo de aplicaciones para móviles con Android",
        trazo, hOffset: 10, vOffset: 40, pincel);
}

```



**Paso 4.** Modifica los parámetros de `canvas.drawTextOnPath()` hasta que comprendas su significado.

```

public void drawTextOnPath (String text, Path path, float
hOffset, float vOffset, Paint paint)

```

Added in [API level 1](#)

Draw the text, with origin at (x,y), using the specified paint, along the specified path. The paint's Align setting determines where along the path to start the text.

#### Parameters

**text** The text to be drawn  
**path** The path the text should follow for its baseline  
**hOffset** The distance along the path to add to the text's starting position

**Paso 5.** ¿Podrías dibujar el texto a favor de las agujas del reloj, pero por fuera del círculo?



```
canvas.drawTextOnPath("Desarrollo de aplicaciones para móviles con Android",
    trazo, 0, -40, pincel);
```



### Parte V: Un ejemplo de Path más complejo

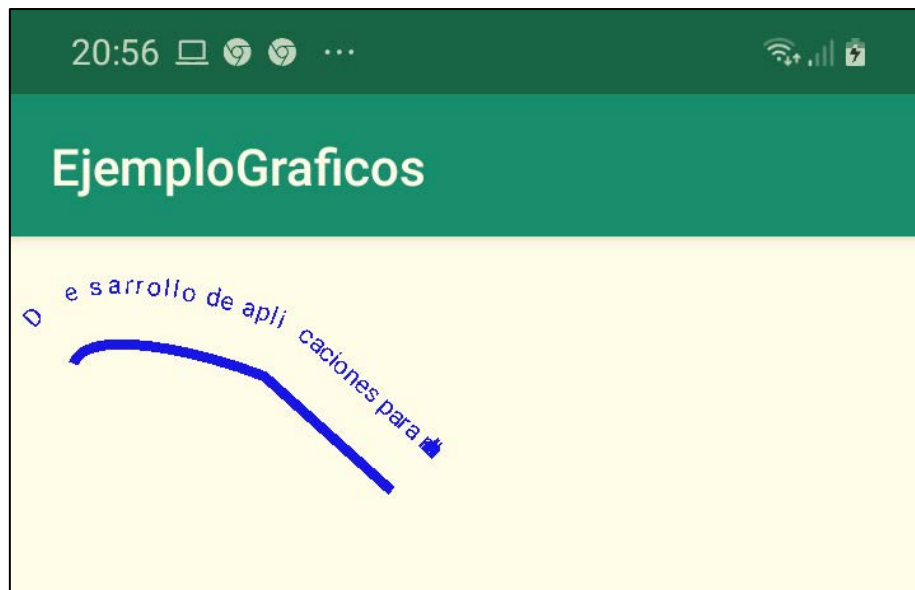
**Paso 1.** Reemplaza las dos primeras líneas del ejemplo anterior por:

```
@Override
protected void onDraw(Canvas canvas) {

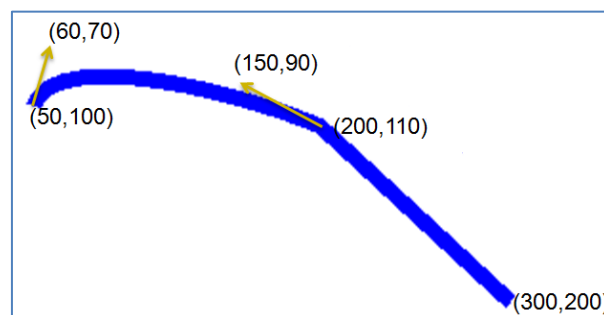
    Path trazo = new Path();
    trazo.moveTo( x: 50, y: 100);
    trazo.cubicTo( x1: 60, y1: 70, x2: 150, y2: 90, x3: 200, y3: 110);
    trazo.lineTo( x: 300, y: 200);

    canvas.drawColor(Color.WHITE);
    Paint pincel = new Paint();
    pincel.setColor(Color.BLUE);
    pincel.setStrokeWidth(8);
    pincel.setStyle(Style.STROKE);
    canvas.drawPath(trazo, pincel);
    pincel.setStrokeWidth(1);
    pincel.setStyle(Style.FILL);
    pincel.setTextSize(20);
    pincel.setTypeface(Typeface.SANS_SERIF);
    canvas.drawTextOnPath( text: "Desarrollo de aplicaciones para móviles con Android",
        trazo, hOffset: 0, vOffset: -40, pincel);
}
```

**Paso 2.** El resultado obtenido ha de ser similar a:



El trazo comienza desplazándose a las coordenadas (50,100). Luego introduce una curva cúbica o Bézier hasta la coordenada (200,110). Una curva Bézier introduce dos puntos de control, el primero (60,70) permite controlar como arranca la dirección del comienzo de la curva y el segundo (150,90) la dirección del final de la curva. Funciona de la siguiente manera, si trazas una recta desde el comienzo de la curva (50,100) hasta el primer punto de control (60,70) la curva se trazará tangencialmente a esta recta. Finalmente, se añade una línea desde las coordenadas (200,110) hasta (300,200).



**Paso 3.** ¿Por qué aparece una separación entre la “p” y la “l”? ¿Qué dos parámetros del siguiente gráfico tendríamos que modificar para que esta separación no estuviera? (Solución: (150,90) => (150,65))

```
@Override
protected void onDraw(Canvas canvas) {
    Path trazo = new Path();
    trazo.moveTo(50, 100);
    trazo.cubicTo(60,70, 150,65, 200,110);
    trazo.lineTo(300,200);
    canvas.drawColor(Color.WHITE);
    Paint pincel = new Paint();
    pincel.setColor(Color.BLUE);
    pincel.setStrokeWidth(8);
    pincel.setStyle(Style.STROKE);
    canvas.drawPath(trazo, pincel);
    pincel.setStrokeWidth(1);
    pincel.setStyle(Style.FILL);
    pincel.setTextSize(20);
    pincel.setTypeface(Typeface.SANS_SERIF);
    canvas.drawTextOnPath("Desarrollo de aplicaciones para móviles con Android",
        trazo, 0, -40, pincel);
}
```

## Ejemplo Graficos

Desarrollo de aplicaciones para n

