

SEMINARIO DE LENGUAJES

OPCIÓN ANDROID



Layouts

Esp. Fernández Sosa Juan Francisco

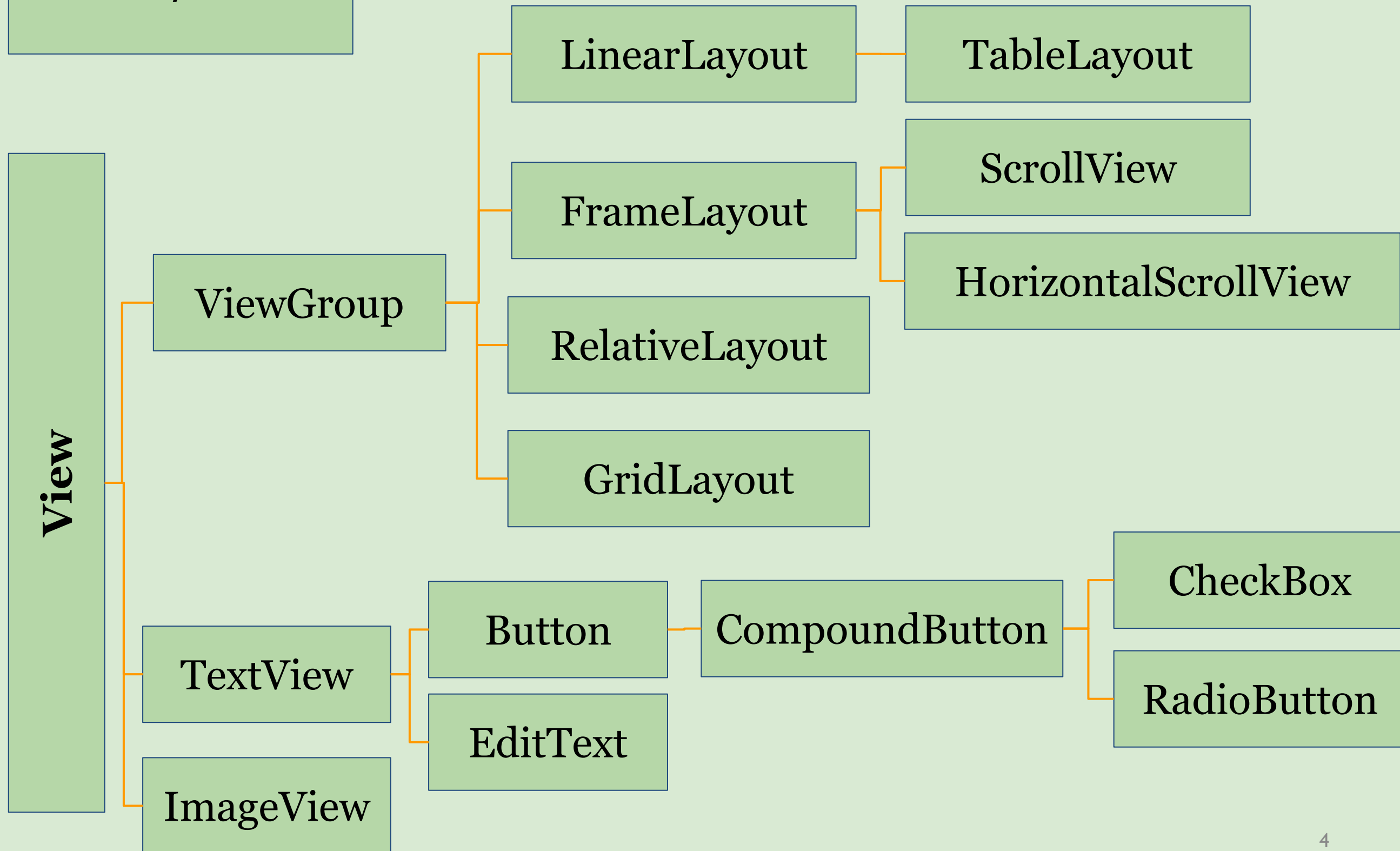
Layouts

- El **Layout** de una **activity** representa el diseño de la interfaz de usuario determinando la disposición de distintos componentes visuales (vistas o **views**) en la misma.
- Los **layouts** también son vistas pero pertenecen a una categoría específica de vistas (**ViewGroup**)

ViewGroups

- Los elementos visuales simples, como lo son los **EditText**, **TextView**, **Button**, etc. son clases particulares de vistas (**View**) que deben ser dispuestos dentro de un contenedor
- El contenedor es un **ViewGroup** que define el modo que se muestran los elementos hijos que aloja.
- Conoceremos algunos de los **ViewGroups** más conocidos...

Algunas Vistas/Views




Layouts

- La interfaz de usuario puede ser definida mediante:
 - **Archivos XML**
 - **Programáticamente desde Kotlin/Java**

Layouts desde archivos XML

- Las **Activities** que definen su interfaz por medio de **archivos XML**, asocian estos archivos mediante la instrucción **setContentView()** en el callback **onCreate**

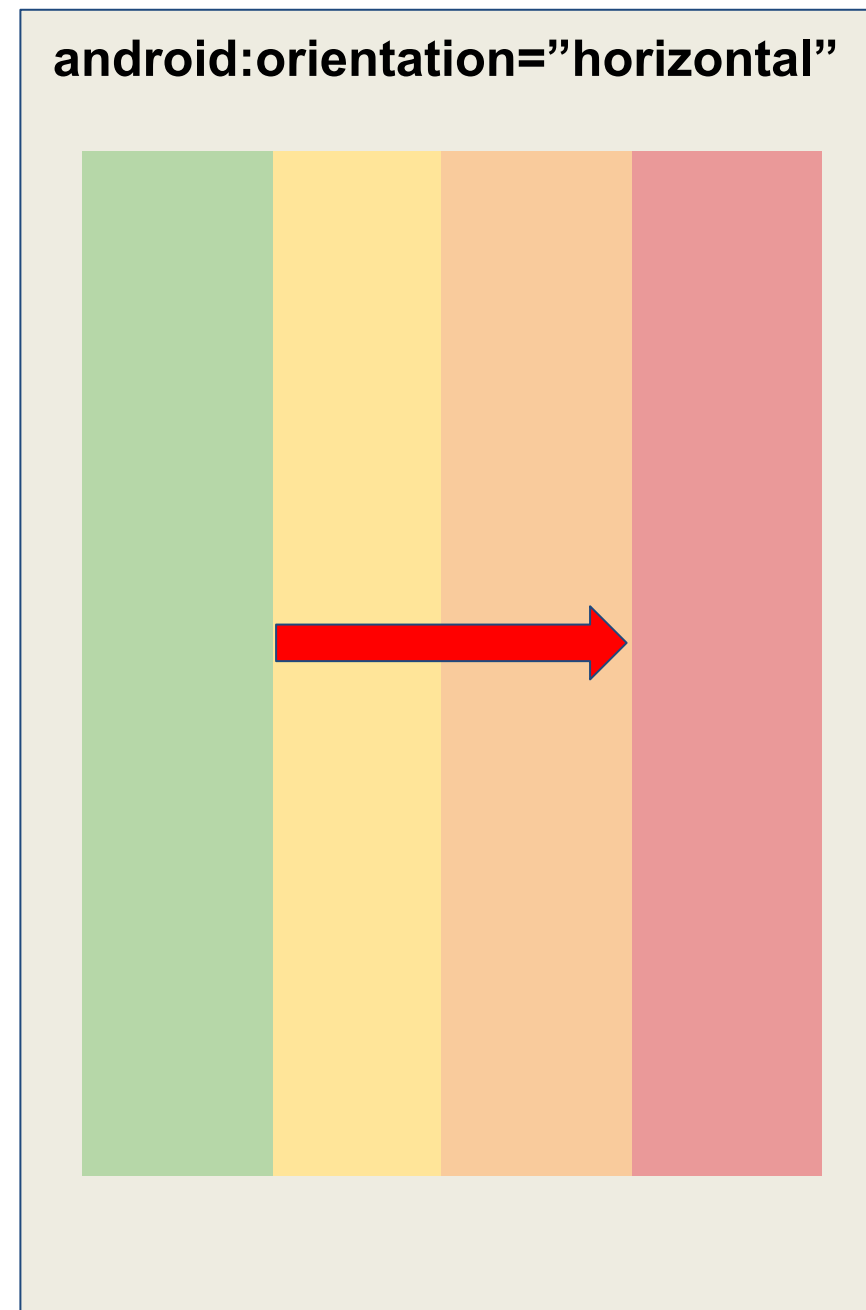
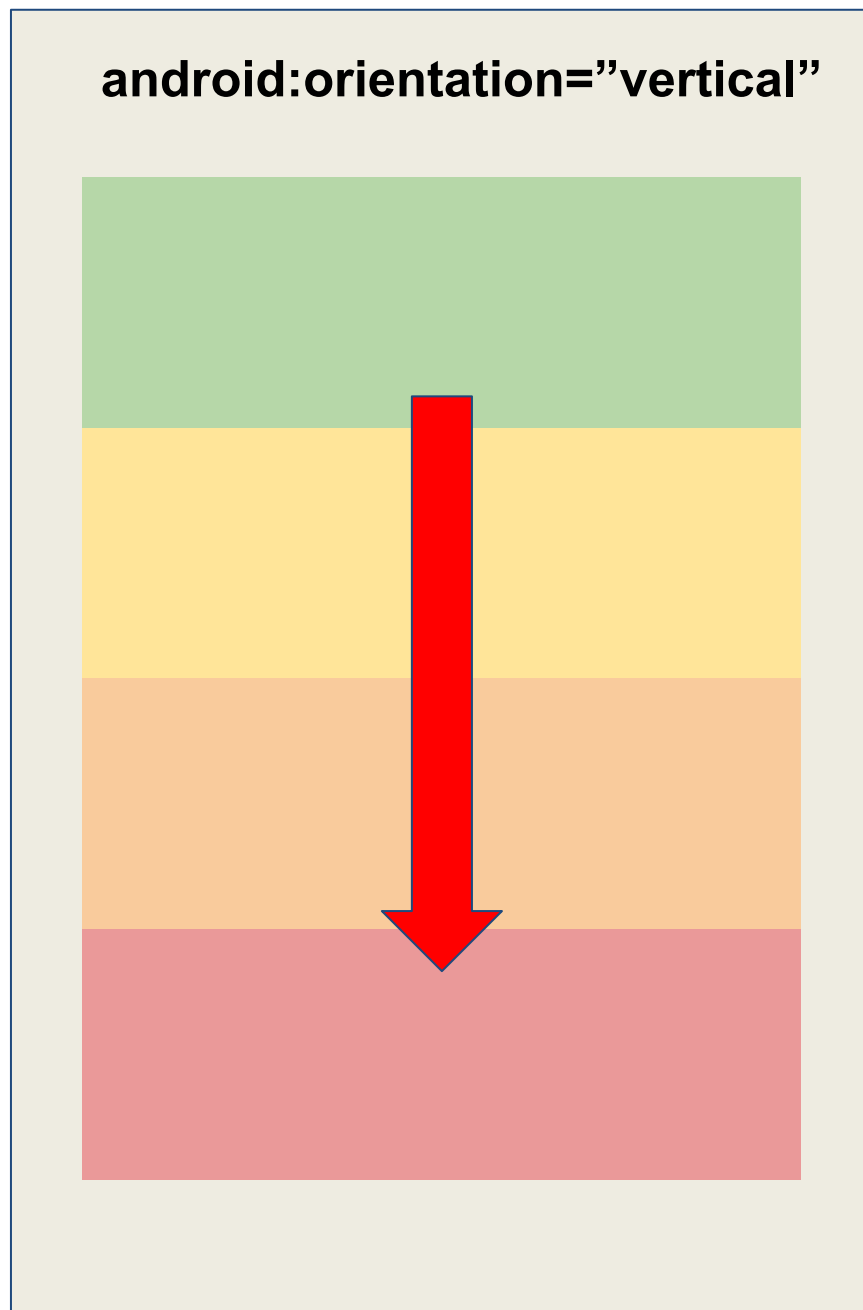
A screenshot of an IDE window showing the MainActivity.kt file. The window has two tabs: 'activity_main.xml' and 'MainActivity.kt'. The 'MainActivity.kt' tab is active, displaying the following Kotlin code:

```
1 package com.example.myapplication
2
3 import ...
4
5
6 class MainActivity : AppCompatActivity() {
7     override fun onCreate(savedInstanceState: Bundle?) {
8         super.onCreate(savedInstanceState)
9         setContentView(R.layout.activity_main)
10    }
11 }
```

The code is color-coded: 'package' is blue, 'import' is blue, 'class' is blue, 'override fun onCreate' is blue, 'super.onCreate' is blue, 'setContentView' is blue, and 'R.layout.activity_main' is purple. The line numbers 1 through 11 are visible on the left side of the editor.

**Vamos a presentar algunos
de los layouts más
populares**

LinearLayout

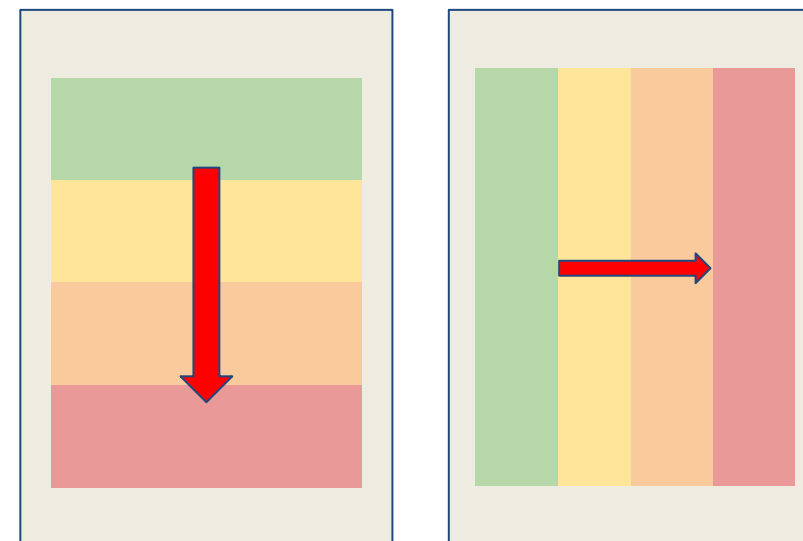


LinearLayout

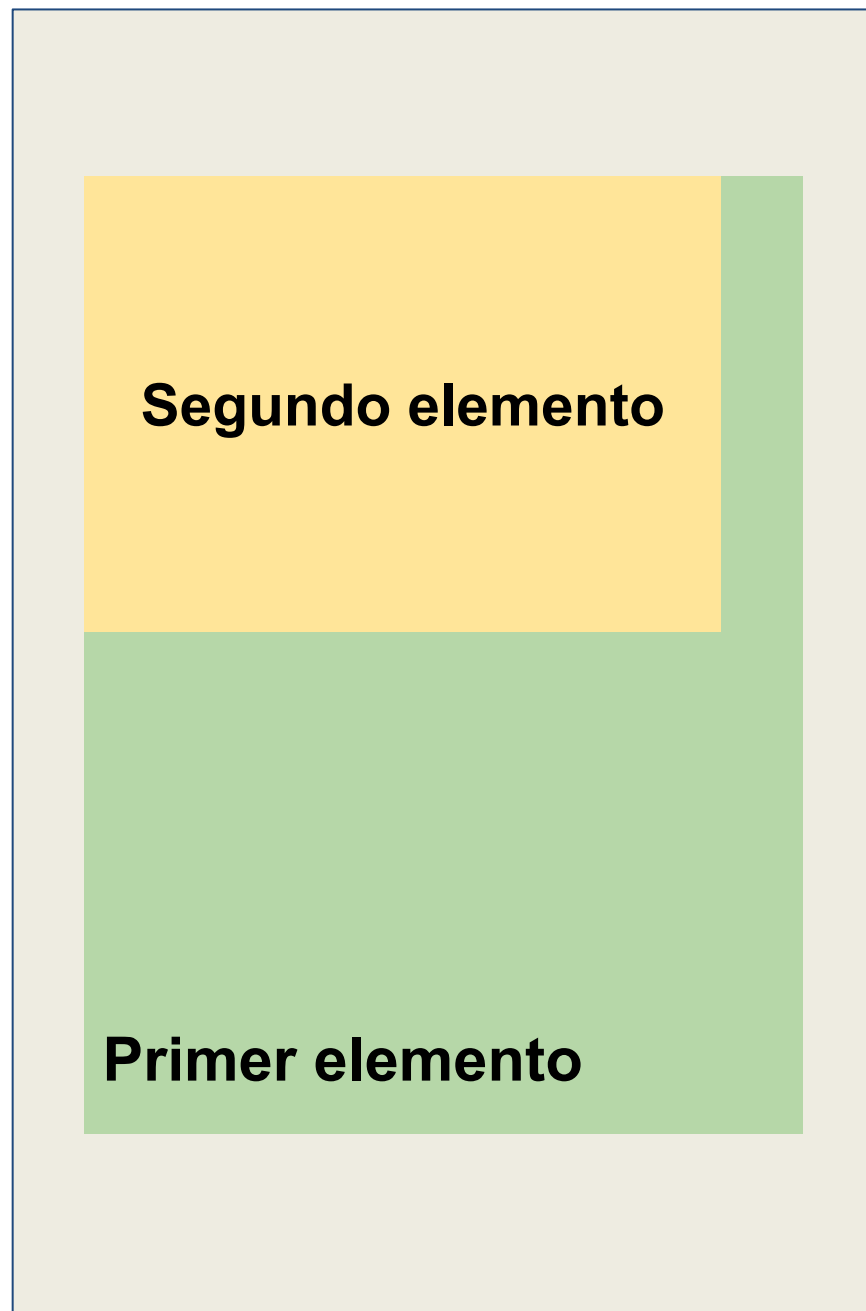
- Es un **ViewGroup** que alinea a los elementos hijos en una única dirección.
 - La dirección puede ser **vertical** u **horizontal**

```
<LinearLayout  
xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout width="match parent"  
    android:layout height="match parent"  
    android:orientation="vertical">
```

```
</LinearLayout>
```



FrameLayout

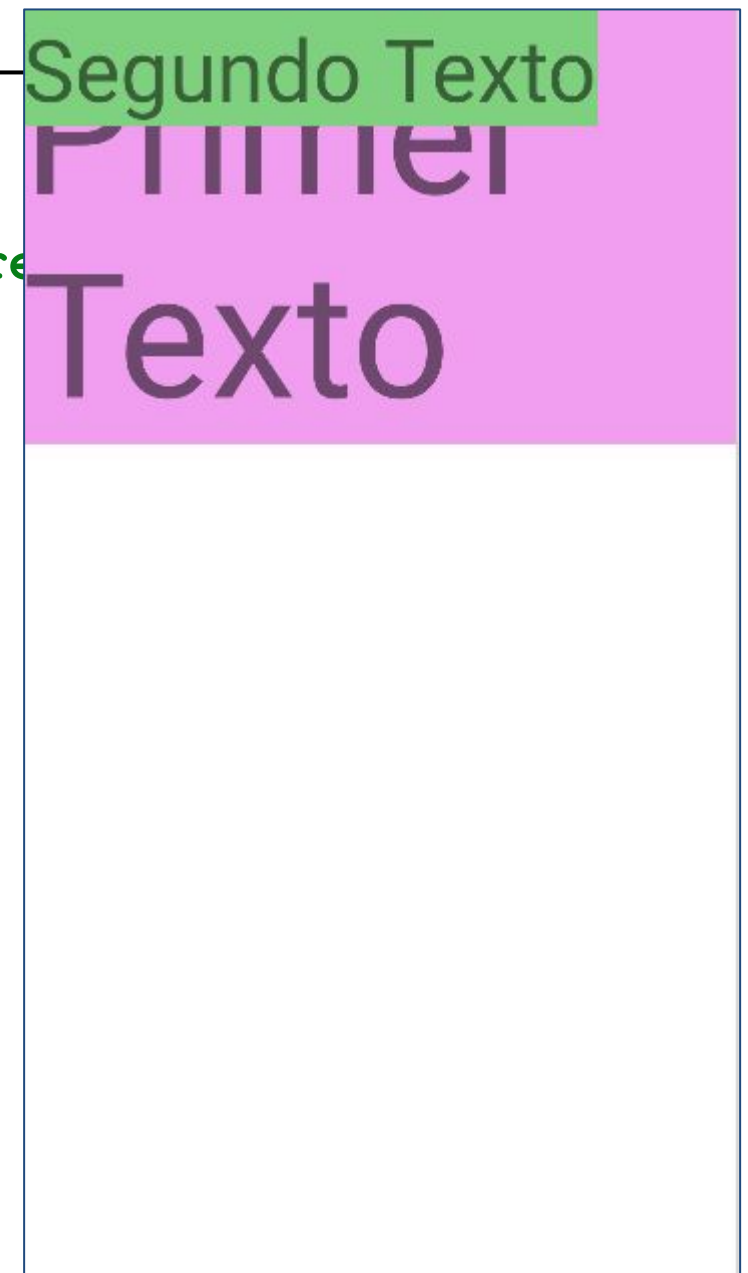


- Un FrameLayout es un ViewGroup simple y eficiente.
- Pensado para ser usado con un **view hijo** o con **Views** que **admiten solapamiento**.
- Los hijos se dibujan apilados, con el más reciente en la cima.

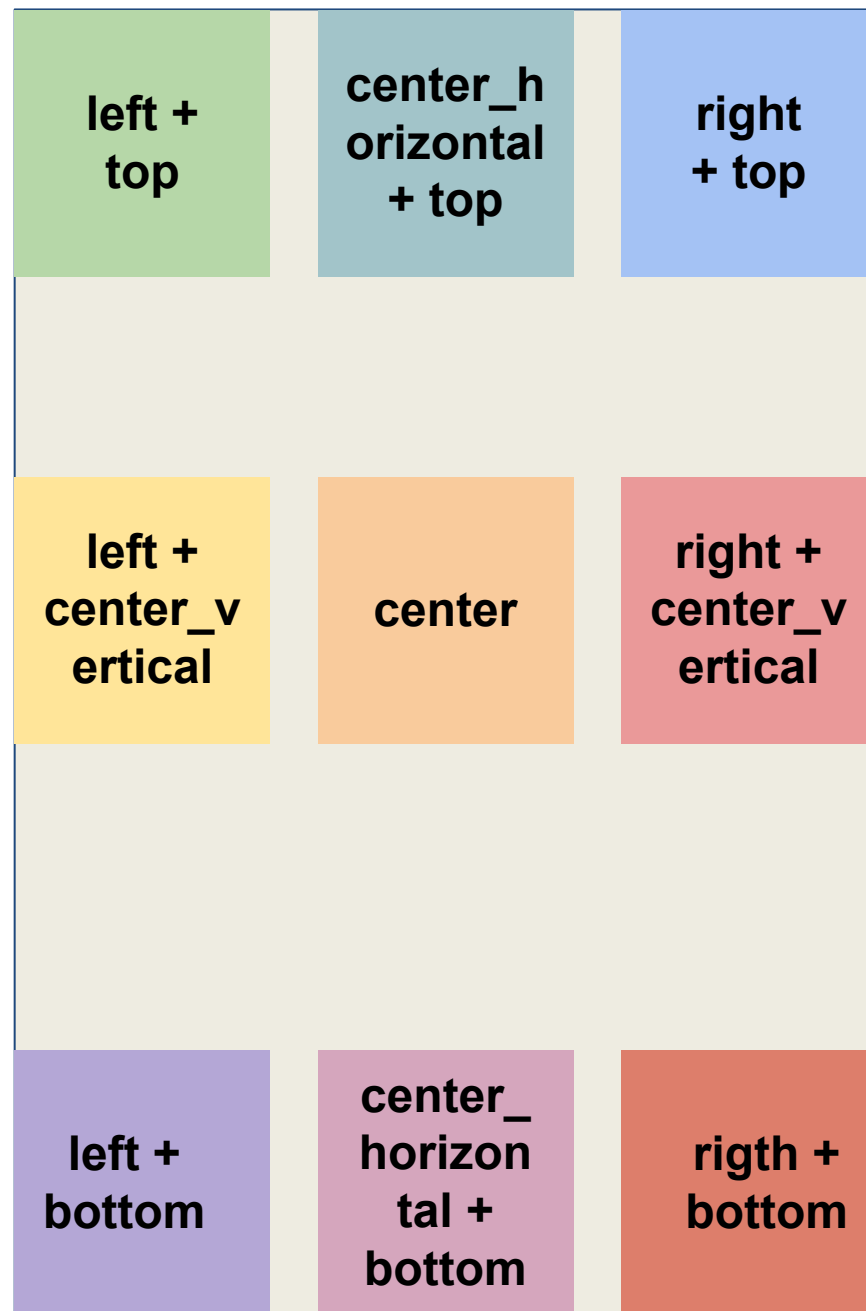
Ejercitación

- Crear un nuevo proyecto nuevo en Android Studio seleccionando el template “**Empty Views Activity**”
- Eliminar el contenido del archivo XML de la actividad generada y reemplazarlo por el siguiente

```
<FrameLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="100dp"
        android:background="#F19EF1"
        android:text="Primer Texto"/>
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="50dp"
        android:background="#7FD17F"
        android:text="Segundo Texto"/>
</FrameLayout>
```



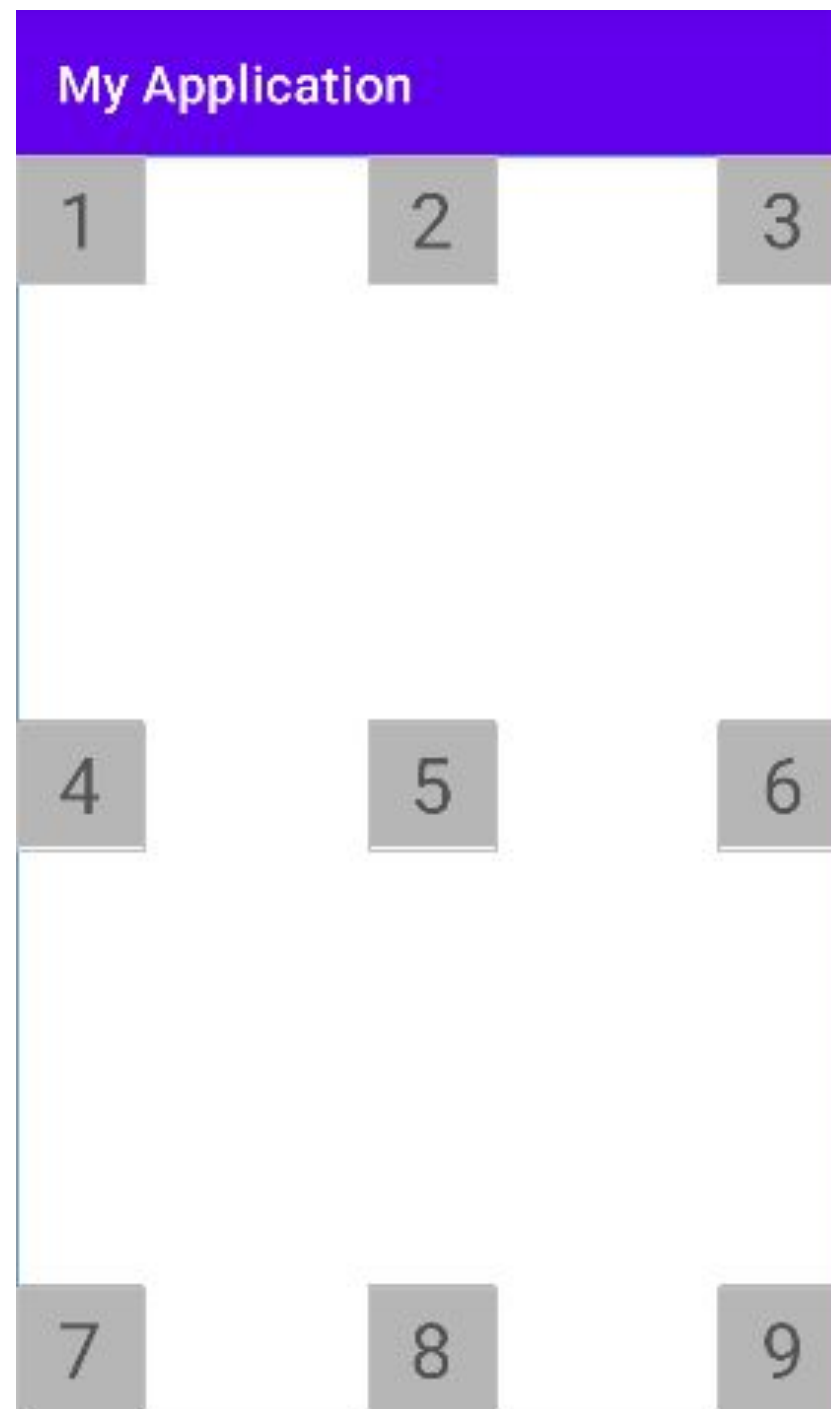
FrameLayout



- Se le puede asignar a cada hijo de un FrameLayout el atributo **layout_gravity**.
- El **layout_gravity** define cómo los componentes se disponen con respecto al FrameLayout. Pueden hacerlo en la parte superior, del centro o inferior.
- Algunos de los valores que puede tomar este atributo son: **center**, **center_horizontal**, **center_vertical**, **top**, **bottom**, **right**, **end**, **start**, **left**.
- Pueden combinarse utilizando el carácter “|”. Por ejemplo “**center|right**”

Ejercitación

- Codificar la siguiente **activity**



**Antes de conocer el tercer
layout, vamos a hacer un
ejercicio...**

Ejercitación

- En el proyecto que están trabajando, **eliminar** todo el **contenido del XML** que define la actividad
- Definir un **LinearLayout** con orientación **vertical**
- Agregar al LinearLayout un **botón** con texto “Botón 1”. Asignarle un **alto de 150dp** y que su **ancho** sea **igual al del padre**
- Copiar y pegar el botón **cuatro veces**, modificando su texto en función al número de botón
- **Ejecutar en el emulador**

¿Se visualizan
adecuadamente todos
los botones?



<ScrollView>
Al rescate...

ScrollView y HorizontalScrollView

- **ScrollView** es un **FrameLayout** especializado que puede hacer **scroll vertical** sobre el elemento que contiene. **Sólo puede alojar un único hijo.**
- **HorizontalScrollView** es un **FrameLayout** especializado que puede hacer **scroll horizontal** sobre el elemento que contiene. **Sólo puede alojar un único hijo.**

ScrollView y HorizontalScrollView

```
activity_main.xml x
1 <LinearLayout
2   xmlns:android="http://schemas.android.com/apk/res/android"
3   android:layout_width="match_parent"
4   android:layout_height="match_parent"
5   android:orientation="vertical">
6   <ScrollView
7     android:layout_width="match_parent"
8     android:layout_height="match_parent">
9     <Button
10       android:layout_width="match_parent"
11       android:layout_height="150dp"
12       android:text="Botón 1"/>
13     <Button
14       android:layout_width="match_parent"
15       android:layout_height="150dp"
16       android:text="Botón 2"/>
17     <Button
18       android:layout_width="match_parent"
19       android:layout_height="150dp"
20       android:text="Botón 3"/>
21     <Button
22       android:layout_width="match_parent"
23       android:layout_height="150dp"
24       android:text="Botón 4"/>
25   </ScrollView>
26 </LinearLayout>
```

Pregunta: ¿Por qué esta solución no podría ser válida?

Respuesta: Porque el **ScrollView** sólo puede almacenar **un hijo directo**

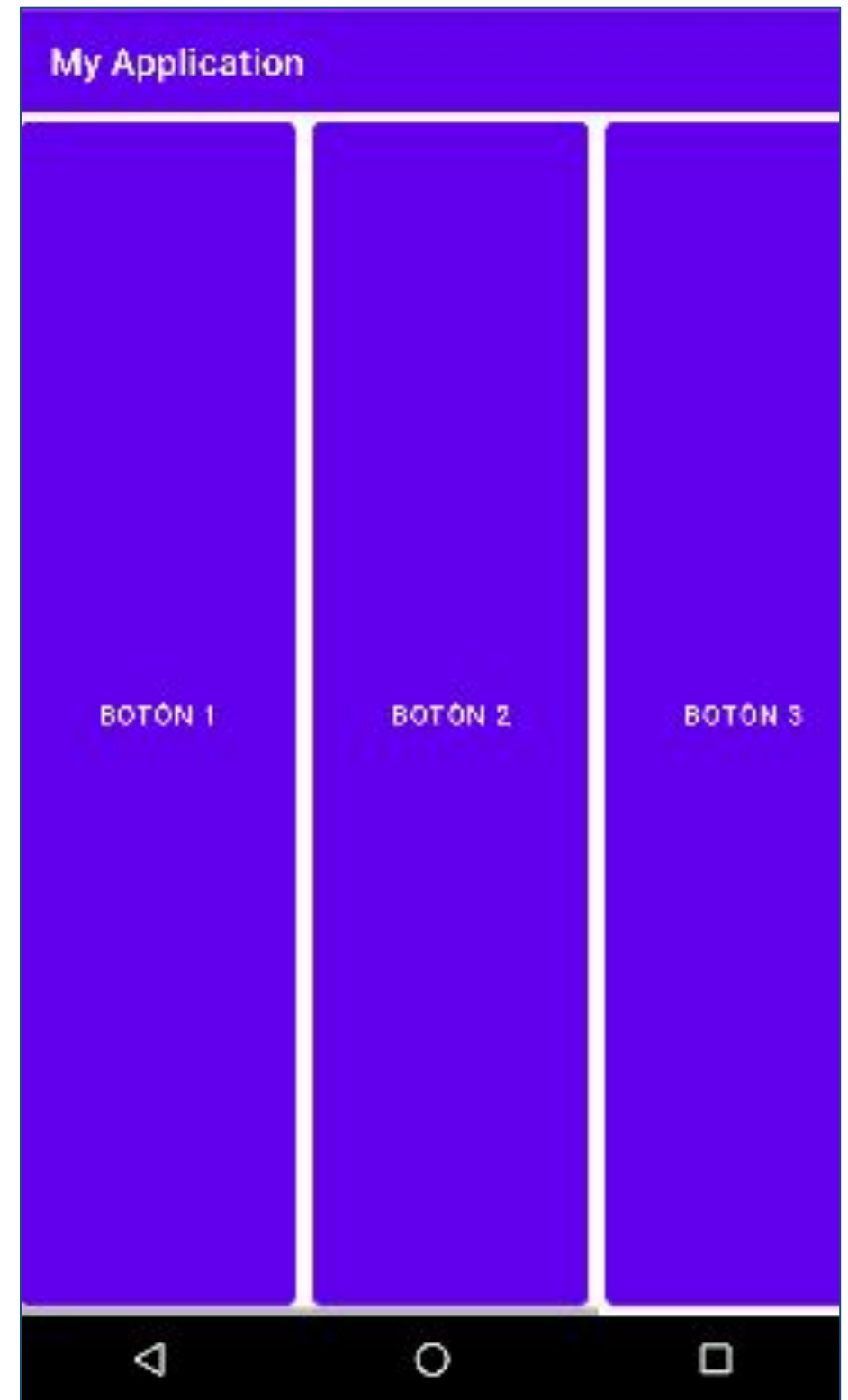
ScrollView y HorizontalScrollView

```
activity_main.xml
1 <ScrollView
2     xmlns:android="http://schemas.android.com/apk/res/android"
3     android:layout_width="match_parent"
4     android:layout_height="match_parent">
5     <LinearLayout
6         android:layout_width="match_parent"
7         android:layout_height="match_parent"
8         android:orientation="vertical">
9         <Button
10             android:layout_width="match_parent"
11             android:layout_height="150dp"
12             android:text="Botón 1"/>
13         <Button
14             android:layout_width="match_parent"
15             android:layout_height="150dp"
16             android:text="Botón 2"/>
17         <Button
18             android:layout_width="match_parent"
19             android:layout_height="150dp"
20             android:text="Botón 3"/>
21         <Button
22             android:layout_width="match_parent"
23             android:layout_height="150dp"
24             android:text="Botón 4"/>
25     </LinearLayout>
26 </ScrollView>
```

Solución válida:
El **ScrollView**
aloja un único hijo
directo: al
LinearLayout
que contiene a los
botones

Ejercitación

- Modificar la aplicación para disponer los botones **horizontalmente**.
- El scroll ahora debe ser horizontal.
- El **ancho** de los botones debe ser de **150dp** y su **altura depende de** la altura del **padre**.
- Establecer la propiedad **layout_marginRight** con el valor **30dp**



Solución

```
activity_main.xml x
1 <HorizontalScrollView
2     xmlns:android="http://schemas.android.com/apk/res/android"
3     android:layout_width="match_parent"
4     android:layout_height="match_parent">
5     <LinearLayout
6         android:layout_width="wrap_content"
7         android:layout_height="match_parent"
8         android:orientation="horizontal">
9         <Button
10             android:layout_marginRight="10dp"
11             android:layout_height="match_parent"
12             android:layout_width="150dp"
13             android:text="Botón 1"/>
14         <Button
15             android:layout_marginRight="10dp"
16             android:layout_height="match_parent"
17             android:layout_width="150dp"
18             android:text="Botón 2"/>
19         <Button
20             android:layout_marginRight="10dp"
21             android:layout_height="match_parent"
22             android:layout_width="150dp"
23             android:text="Botón 3"/>
24         <Button
25             android:layout_marginRight="10dp"
26             android:layout_height="match_parent"
27             android:layout_width="150dp"
28             android:text="Botón 4"/>
29     </LinearLayout>
30 </HorizontalScrollView>
31
```

Solución: Se debe utilizar un
<HorizontalScrollView>

**Vamos a acceder
programáticamente a los
elementos visuales (**views**)
de la activity**

Para **identificar a los **views**
del layout vamos a utilizar
el **atributo id****

Atributo ID

.XML

```
<Button
    android:id="@+id/boton1"
    android:layout_marginRight="10dp"
    android:layout_height="match_parent"
    android:layout_width="150dp"
    android:text="Botón 1"/>
```

El **id** es un atributo especial que permite **identificar al elemento XML desde el código Kotlin**

.kt

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
    var b = findViewById<Button>(R.id.boton1)
}
```


Atributo ID

```
<HorizontalScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="match_parent"
        android:orientation="horizontal">
        <Button
            android:id="@+id/boton1"
            android:layout_marginRight="10dp"
            android:layout_height="match_parent"
            android:layout_width="150dp"
            android:text="Botón 1"/>
        <Button
            android:id="@+id/boton2"
            android:layout_marginRight="10dp"
            android:layout_height="match_parent"
            android:layout_width="150dp"
            android:text="Botón 2"/>
    </LinearLayout>
</HorizontalScrollView>
```

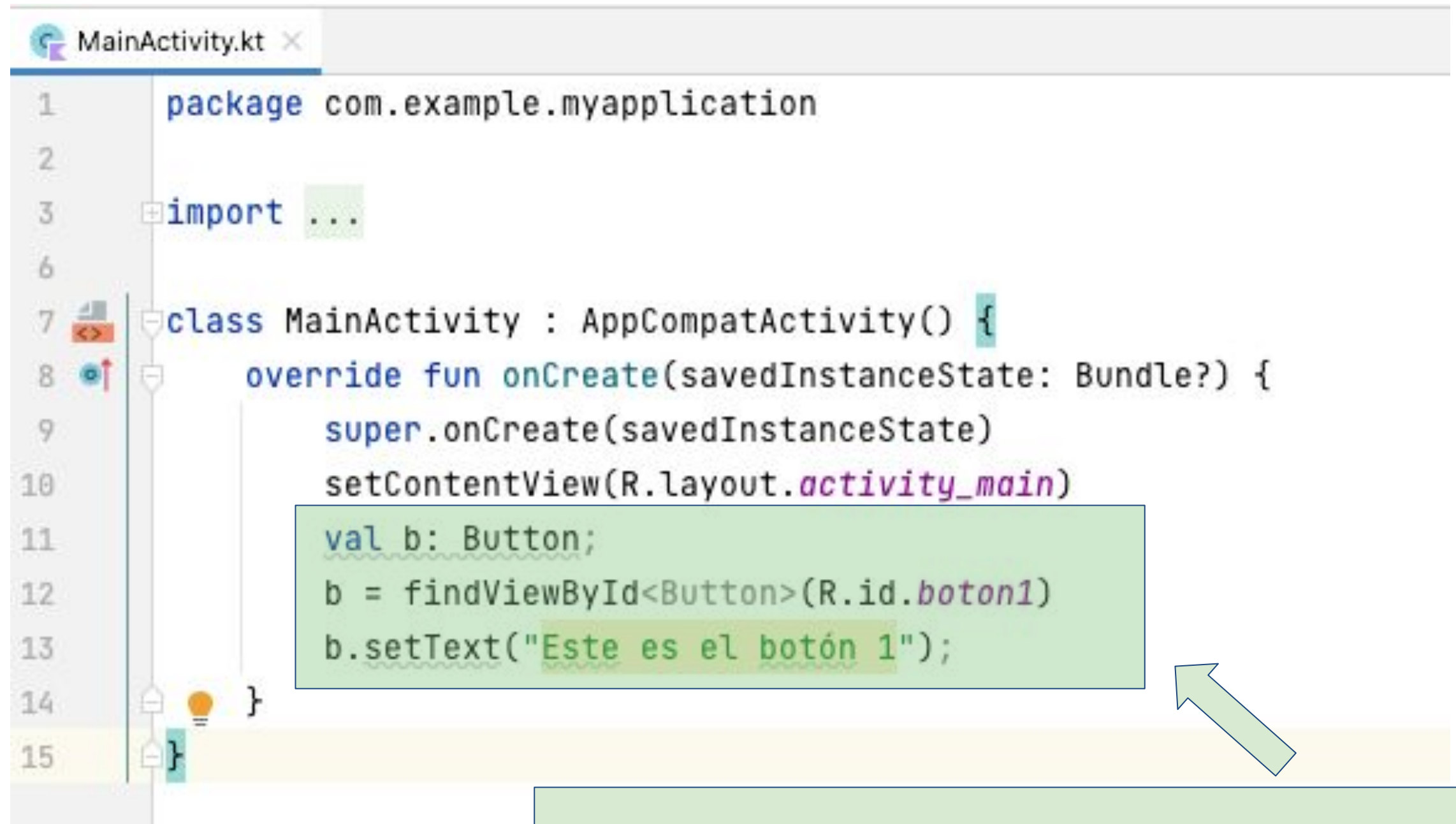
**Agregar el atributo
android:id a cada uno
de los botones**

¿Qué significa “@+id/boton1”?

- Al tipear **@+id/** el entorno convierte una etiqueta en un recurso con un nombre determinado.
- De esta manera, al tipear “**@+id/boton1**” se **crea un recurso** llamado **boton1** que luego **puede referenciarse** desde el código Kotlin por medio de la clase estática **R.id**
- La clase estática **R** y sus clases miembros anidadas se generan automáticamente cuando se compila la aplicación.
- En versiones más recientes, el plugin de Gradle genera el **archivo de bytecode R.class** directamente en lugar del **archivo R.java**.

```
# static fields
.field public static final boton1:I = 0x7f08005d
.field public static final boton2:I = 0x7f08005e
.field public static final boton3:I = 0x7f08005f
.field public static final boton4:I = 0x7f080060
```


Ejercitación



```
1 package com.example.myapplication
2
3 import ...
4
5
6
7 class MainActivity : AppCompatActivity() {
8     override fun onCreate(savedInstanceState: Bundle?) {
9         super.onCreate(savedInstanceState)
10        setContentView(R.layout.activity_main)
11        val b: Button;
12        b = findViewById<Button>(R.id.boton1)
13        b.setText("Este es el botón 1");
14    }
15 }
```

Agregar estas tres instrucciones al método **onCreate()** y ejecutar en el emulador para comprobar el resultado.

Ejercitación

```
MainActivity.kt x
1 package com.example.myapplication
2
3 import ...
6
7 class MainActivity : AppCompatActivity() {
    instanceState:
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
    val b: Button;
    b = findViewById<Button>(R.id.boton1)
    b.setText("Este es el botón 1");
}
```

Se define una constante “b” del tipo **Button**

Se asigna a la constante **b** el objeto **View** de la actividad cuyo id es **boton1** (el método **findViewById** retorna un objeto tipo **View**, es por eso que se debe castear a **Button**)

Se establece el **texto** del **botón**

My Application

ESTE ES EL
BOTÓN 1

BOTÓN 2