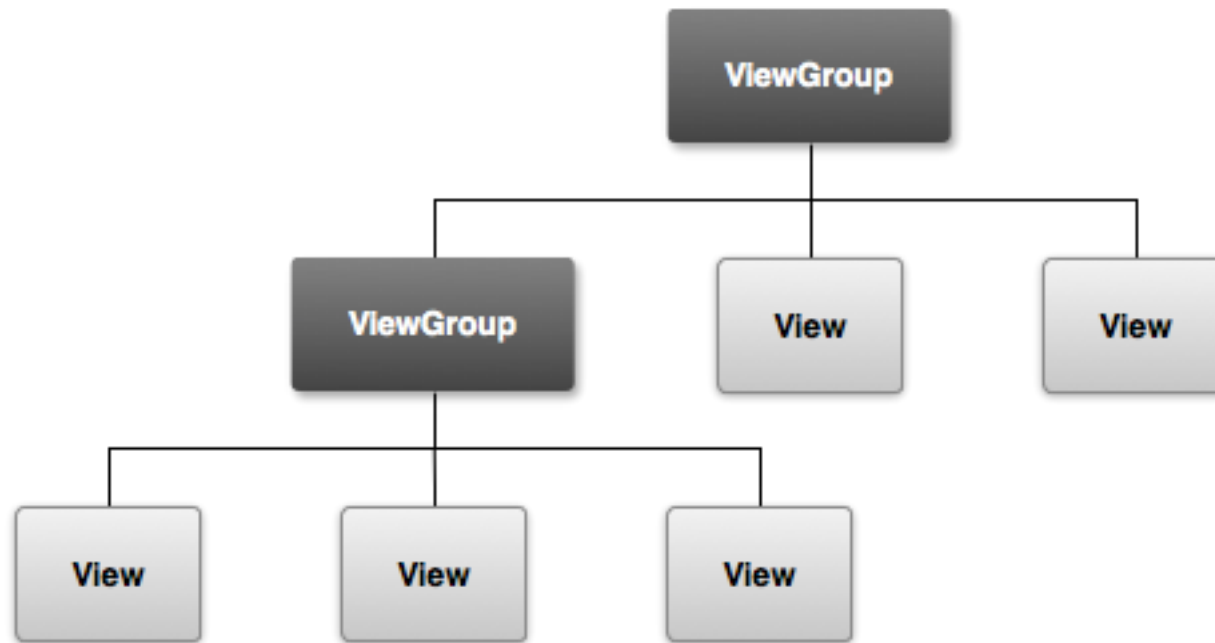


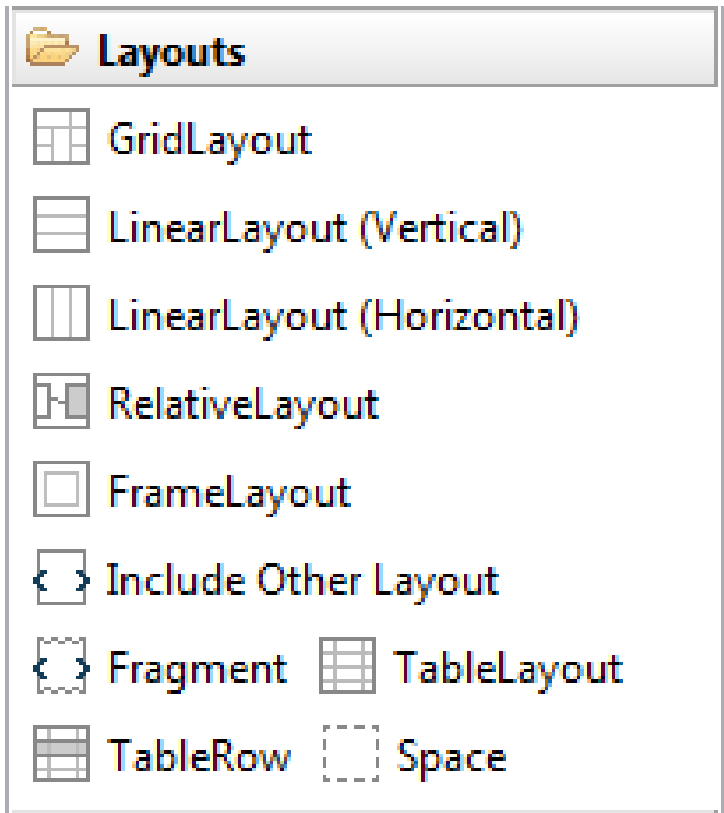
# Android: Layouts

Interfaz de usuario

# Jerarquía de ViewGroups y Views



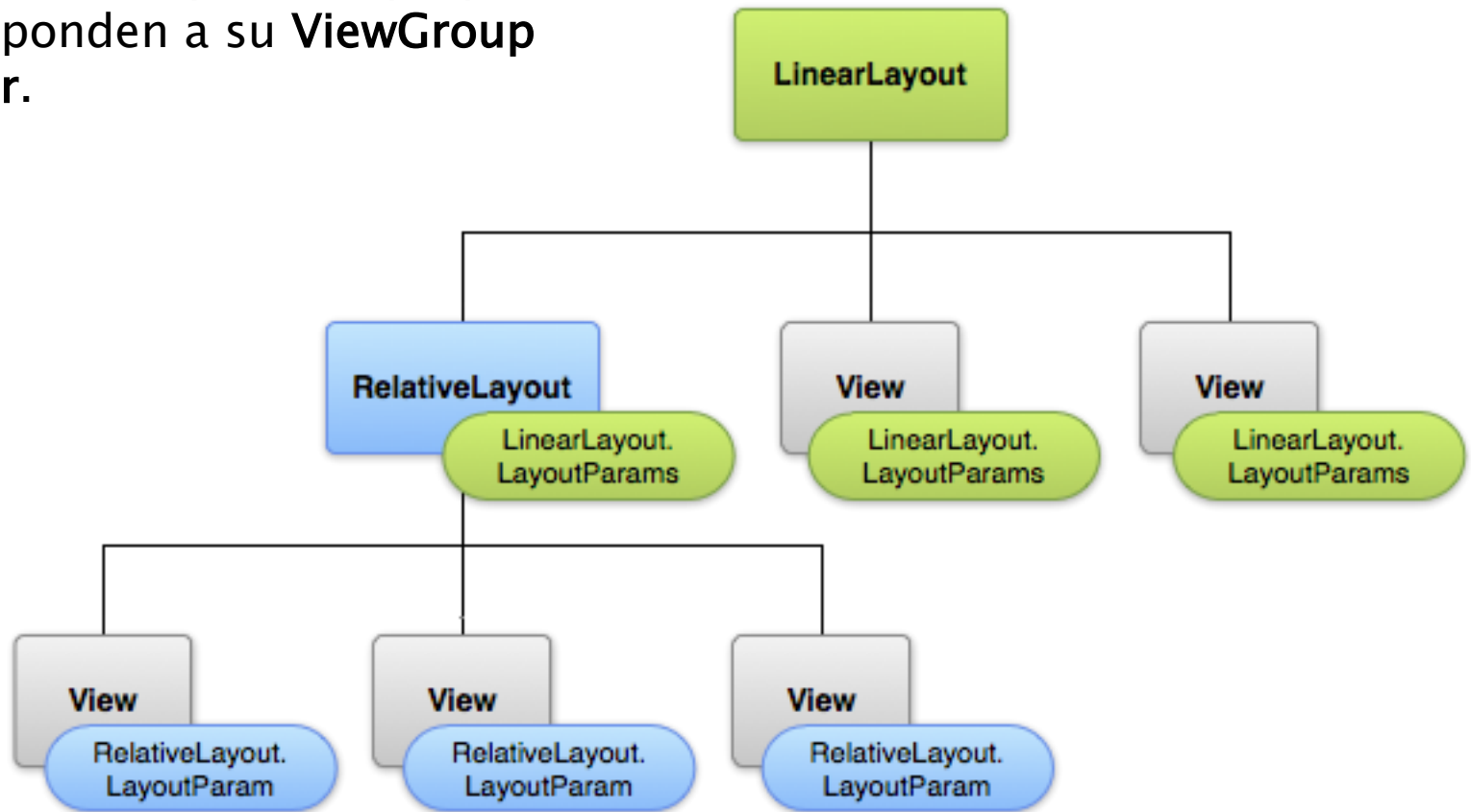
# Layouts



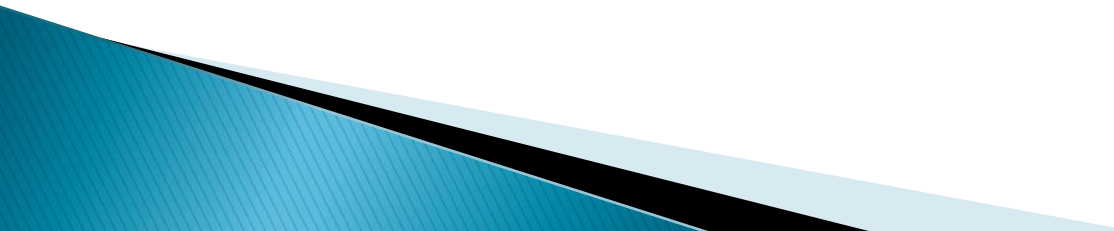
- Cada pantalla es un **layout** o diseño.
- Los layouts se almacenan en la carpeta **/res/layout**.
- Una pantalla tiene un solo elemento raíz de tipo **ViewGroup**.
- Los ViewGroups más utilizados son **LinearLayout** y **RelativeLayout**.
- El LinearLayout puede disponer su contenido de forma **vertical** u **horizontal**.
- Dentro del layout principal podremos insertar tantos **ViewGroups** y **Views** como sean necesarios.
- Dentro de cualquier elemento de tipo **ViewGroup** podremos a su vez insertar más elementos.
- Los elementos de tipo **View** siempre serán elementos finales.

# Ámbito de un layout


A los elementos de tipo View y ViewGroup se les aplica las propiedades que corresponden a su **ViewGroup** contenedor.



# LinearLayout

- ▶ En el de tipo **vertical**, se coloca un elemento encima de otro, de arriba abajo.
  - ▶ En el de tipo **horizontal**, se coloca un elemento al lado de otro, de izquierda a derecha.
  - ▶ Se pueden realizar composiciones más elaboradas insertando unos **LinearLayouts** dentro de otros.
  - ▶ Usando unas pocas **propiedades** se puede mejorar los diseños: `gravity`, `layout_gravity`, `layout_weight`, `layout_width`, `layout_height`.
- 


# RelativeLayout

- ▶ Permite realizar la composición de una pantalla usando **menos elementos** que con LinearLayout.
  - ▶ Al colocar un elemento dentro de este contenedor, siempre se debe especificar su posición de **forma relativa** a otro componente.
  - ▶ Si **cambio la posición** de un componente, se cambia también la de los elementos que se han posicionado de forma relativa a él.
  - ▶ Si **cambio el id** de un componente, se pierde la posición de los elementos que se han colocado de forma relativa a él.
- 

# TableLayout

- ▶ Se utiliza para colocar elementos de forma tabular.
- ▶ Para cada fila nueva se necesita insertar un ViewGroup de tipo **TableRow**.
- ▶ Cada fila tiene tantas **columnas** como elementos.
- ▶ El **ancho de una columna** será el ancho máximo de esa columna de todas las filas.
- ▶ La propiedad **layout\_span** permite agrupar varias columnas dentro de una fila.

Para fundir dos columnas, se hace en el xml. Propiedad **layour\_span** en el objeto.

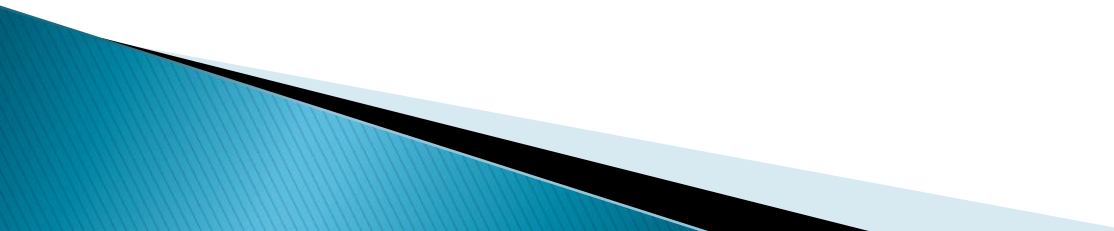


# Unidades de medida

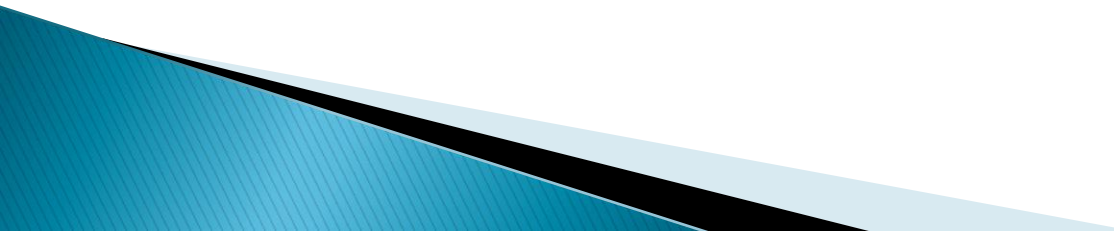
- ▶ **match\_parent**, establece las mismas dimensiones que las del contenedor, se introdujo a partir de la API 8 para sustituir **fill\_parent**;
- ▶ ~~**fill\_parent**~~, establece las mismas dimensiones que las del contenedor;
- ▶ **wrap\_content**, establece las dimensiones necesarias para que quepa el contenido del elemento;
- ▶ **dp** o **dip**, pixel independiente de la densidad, se usa para especificar dimensiones;
- ▶ **sp**, pixel dependiente de la escala, se utiliza para especificar el tamaño de la fuente.



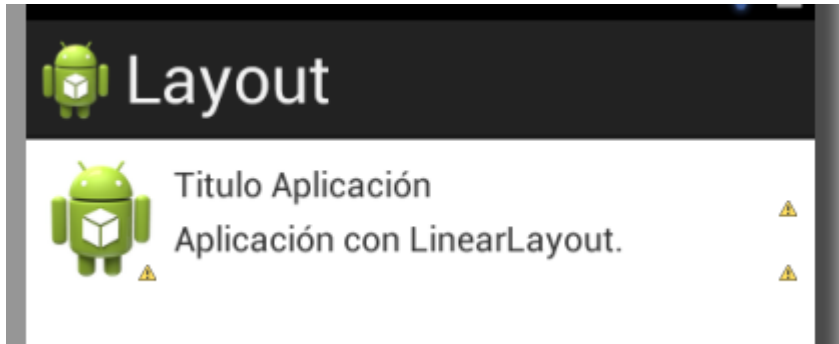
# Elementos de diseño especiales

- ▶ **ListView**, muestra una lista de elementos, cuyo número inicial puede ser desconocido, uno debajo de otro.
  - ▶ **GridView**, muestra una lista de elementos, cuyo número inicial puede ser desconocido, de forma tabular.
  - ▶ **ScrollView**, permite desplazar verticalmente contenido que no cabe en su espacio.
  - ▶ **HorizontalScrollView**, permite desplazar horizontalmente contenido que no cabe en su espacio.
- 

# Elementos de diseño avanzados

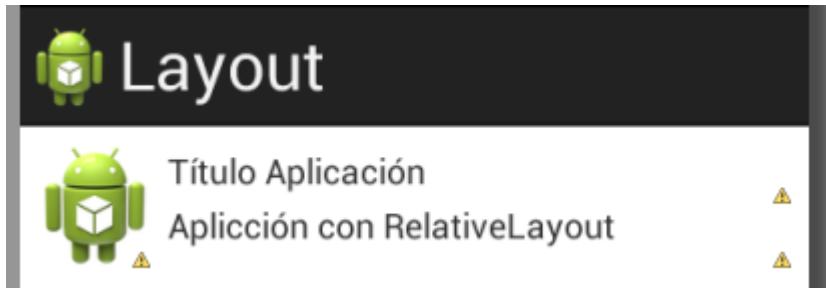
- ▶ Los **fragmentos** son trozos de diseño de pantallas que además pueden tener su comportamiento predefinido.
  - ▶ Las **pestañas** son recursos gráficos mediante los que podemos mostrar diferentes contenidos en la misma pantalla.
  - ▶ Al diseñar un layout podemos crear diseños específicos para el tipo de **orientación**, el **tamaño de pantalla** y otros criterios para conseguir el diseño perfecto para cada pantalla.
- 

# Ejemplo LinearLayout



```
<LinearLayout >  
  <ImageView />  
  <LinearLayout>  
    <TextView android:text="Titulo Aplicación" />  
    <TextView android:text="Aplicación con LinearLayout." />  
  </LinearLayout>  
</LinearLayout>
```

# Ejemplo RelativeLayout

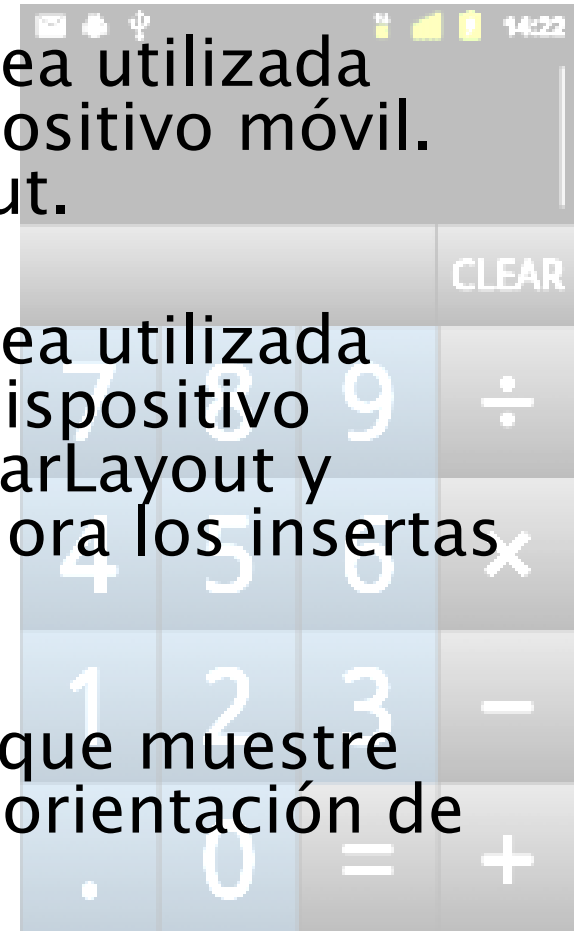


```
<RelativeLayout>  
    <ImageView />  
    <TextView />  
    <TextView />  
</RelativeLayout>
```

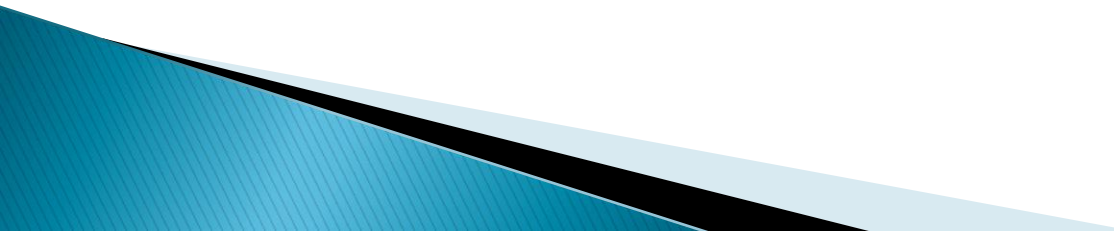
# Ejercicio primero

Los layout deben llamarse igual. Se le indica al crear el layout si es vertical u horizontal.

- ▶ Diseña una calculadora para que sea utilizada con la orientación vertical del dispositivo móvil. Utiliza exclusivamente LinearLayout.
- ▶ Diseña una calculadora para que sea utilizada con la orientación horizontal del dispositivo móvil. El layout principal será LinearLayout y todos los elementos de la calculadora los insertas en un TableLayout.
- ▶ Diseña una aplicación en Android que muestre ambos layouts dependiendo de la orientación de la calculadora.



# Ejercicio segundo

- ▶ Diseña una aplicación en Android que siempre se muestre en orientación vertical, aunque se gire el dispositivo móvil. La aplicación debe disponer de un menú de opciones que permita elegir entre dos layouts diferentes. Al seleccionar cada una de las opciones del menú se mostrará uno de los siguientes layouts.
- 

# Ejercicio segundo



Imagen

Texto con Scroll



Caja o layout

Imagen

La parte de abajo tiene un ScrollView.

Utiliza los layouts que necesites. Investiga.

# Ejercicio segundo

- ▶ Las opciones del menú se crean en /res/menu.

Las opciones son layout vertical y layout horizontal. Se accede al menu con el boton menu.

- ▶ Para visualizar el menú:

```
public boolean onCreateOptionsMenu(Menu menu) {  
    getMenuInflater().inflate(R.menu.archivomenu, menu);  
    return true;  
}
```



# Ejercicio segundo

- Para responder a la opción seleccionada:

@Override

```
public boolean onOptionsItemSelected(MenuItem item) {  
    switch (item.getItemId()) {  
        case R.id.opcionmenu1:  
            ...  
            return true;  
        case R.id.opcionmenu2:  
            ...  
    }  
    return super.onOptionsItemSelected(item);  
}
```

Ponerle evento a cada opción del menu para que haga algo.

Equivalente al break

# Ejercicio segundo

En el menu de ambos ejercicios, incluir un "about" con el nombre y apellidos.

- ▶ Para cambiar el layout:

```
setContentView(R.layout.nombrediseño);
```

- ▶ Para no cambiar la orientación, se indica en el manifiesto para la actividad relacionada:

```
<activity  
    android:name= ".NombreActividad"  
    android:screenOrientation= "portrait"  
    android:label= "@string/cadenatitulo" >
```

siempre horizontal: landscape.  
para que elija el no se pone nada.

# Enlaces

- ▶ Interfaz de usuario:

<http://developer.android.com/guide/topics/ui/index.html>

- ▶ Recursos de una aplicación:

<http://developer.android.com/guide/topics/resources/index.html>

- ▶ Recursos gráficos:

<http://developer.android.com/guide/topics/graphics/index.html>