# Programación para Dispositivos Móviles(T.E.)

## Ciclo I – 2020

**Clase:005**

## Coordinador:Ing. César González





OBJECTIVE-C

# Unidad II

# Interfaz Grafica (continuación ...)

Basado en clases de
Ing. Carlos A. Aguilar
http://developer.android.com/training/basics/firstapp/starting-activity.html

# Agenda

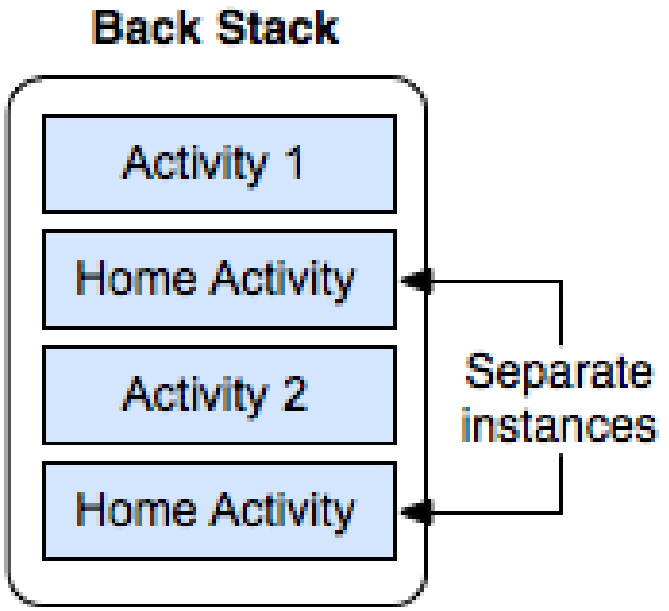
- Desarrollo de Apps para Android
  - Tareas
  - Inicio de Activities(Intents)
  - UI
  - Layouts
  - Eventos

# Tareas(Tasks)

Las tareas o Tasks son grupos de actividades que en conjunto realizar un trabajo determinado. Una tarea es un stack de activades.

Toda una tarea puede pasar al background(cuando el usuario presiona la tecla de Home, utiliza el task manager o abre una notificación)

- Una actividad puede ser instanciada muchas veces, a través de tareas distintas.



# Mensajes (Toast)

```
Context context = getApplicationContext();
CharSequence text = "Hello toast!";
int duration = Toast.LENGTH_SHORT;

Toast toast = Toast.makeText(context, text, duration);
toast.show();
```

# Iniciar una Activity

Las actividades se inician por medio de Intents que describen la actividad que queremos iniciar. Existen 2 formas básicas de iniciar un actividad.

- **Explícitamente**:

Activities dentro de nuestra aplicación

- **Implícitamente**

Describimos al sistema operativo la acción que queremos realizar y Android lanza la actividad que pueda procesar esa acción. Si existe mas de una actividad que cumpla los requisitos, el usuario escoge cual utilizar.

# Iniciar una Activity

Declaramos un Intent, cuyo contexto es la Actividad actual y especificamos la clase de la actividad que deseamos iniciar.

```
Intent intent = new Intent(this, SecondActivity.class);
startActivity(intent);
```

# Iniciar una Activity y capturar el resultado

Muchas veces una activity puede regresar datos a la actividad que la Inicio.

- Para solicitar de que la actividad generadora procese la información devuelta por una actividad hija, la actividad debe de ser llamada usando startActivityForResult()
- Los resultados son capturados asincrónicamente, por medio del callback (es un fragmento de código que se llama después de que un determinado evento ha ocurrido)onActivityResult()

# Iniciar una Activity Interna

```
public void onButtonClick(View view) {

    Intent intent = new Intent(this, SecondActivity.class);
    //Envio de datos
    String message = editText.getText().toString();
    intent.putExtra(EXTRA_MESSAGE, message);

    startActivity(intent);
}
```

Declaramos un intent, en base a un Activity creado previamente(secundario). Agregamos información adicional a través de Extras si son necesarios. Programamos la recepcion de los extras en el activity secundario. Iniciamos la Activity.

# Iniciar una Activity Interna o Externa

```
public void onButtonClick(View view) {

    Intent intent = new Intent(this, SecondActivity.class);
    //Envio de datos
    String message = editText.getText().toString();
    intent.putExtra(EXTRA_MESSAGE, message);

    startActivity(intent);
}
```

Declaramos un intent, en base a un Action. Agregamos información adicional a través de Extras si son necesarios. Iniciamos la Activity

# UI

En Android la interfaz grafica es construida a partir de 2 elementos: Views y Viewgroups

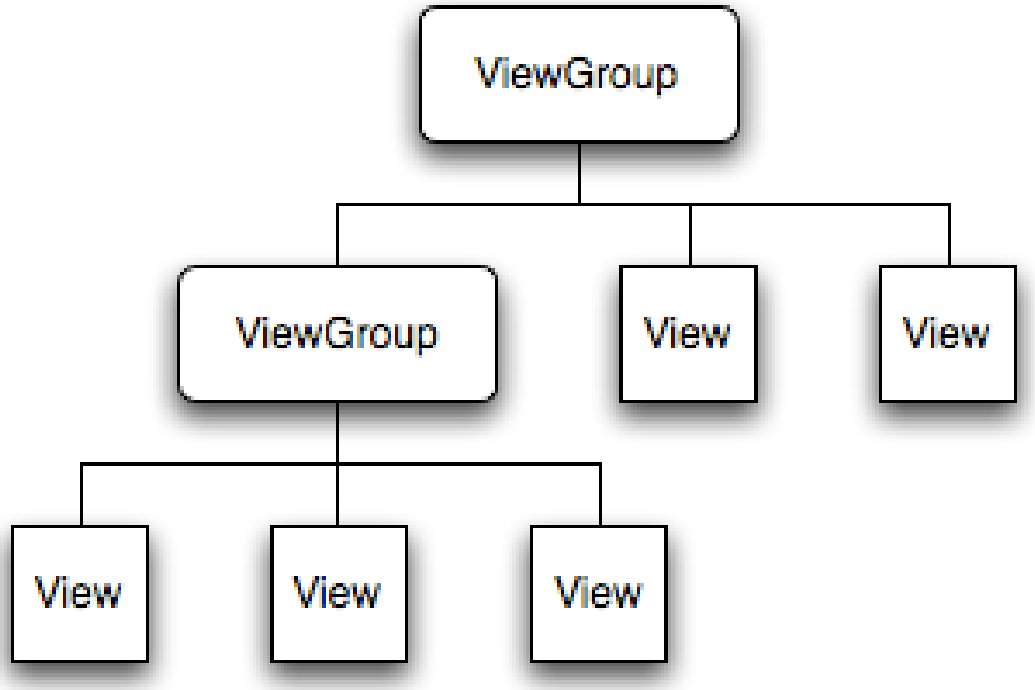
- Views
  - Son las unidades básicas de una interfaz grafica
  - Es la clase base para la los Widgets(Controles)
- ViewGroups
  - Proveen el medio para agrupar views y ordenarlas
  - Son la clase base de los layouts(LinearLayout, ScrollableLayout, etc.)
  - Un ViewGroup sigue siendo un View, excepto que posee la capacidad de contener mas Views

# UI

En Android la interfaz grafica es construida a partir de 2 elementos: Views y Viewgroups

- Views
  - Son las unidades básicas de una interfaz grafica
  - Es la clase base para la los Widgets(Controles)
- ViewGroups
  - Proveen el medio para agrupar views y ordenarlas
  - Son la clase base de los layouts(LinearLayout, ScrollableLayout, etc.)
  - Un ViewGroup sigue siendo un View, excepto que posee la capacidad de contener mas Views

# UI




- La interfaz grafica es organizada en un árbol de elementos.
- La UI es dibujada desde la base del árbol hacia abajo.

# Layouts

La forma mas simple de expresar un layout es a través de XML.

- El nombre de un elemento de XML corresponde al nombre de la clase del elemento que deseamos agregar
- Dentro de un layout solo puede existir un elemento raíz(Un Viewgroup)
- Podemos agregar mas ViewGroups dentro de otro.
- Los layouts son llamados por la actividad por el metodo setContentView()

# Atributos de los elementos un Layout XML - ID

- Es común que exista una correspondencia casi directa entre los atributos de los Views y Viewgroups declarados en XML y los métodos de las clases. Solo es necesario acostumbrarse a la nomenclatura
- Por ejemplo un TextView tiene un atributo text. En la clase de TextView si deseamos cambiar el texto llamamos a setText()

# Atributos de los elementos un Layout XML - ID

- Cualquier View o ViewGroup en Android puede tener un ID para identificar al elemento dentro del Árbol
- Un ID es un entero, que puede ser asignado por el SDK durante la compilación o por el programador
- El programador normalmente agregara una cadena de texto en el atributo id de el elemento y el SDK le asignara un entero automáticamente.

# Atributos de los elementos un Layout XML - ID

La propiedad ID debe de especificarse de la siguiente forma:

**android:id="@+id/my_button”**

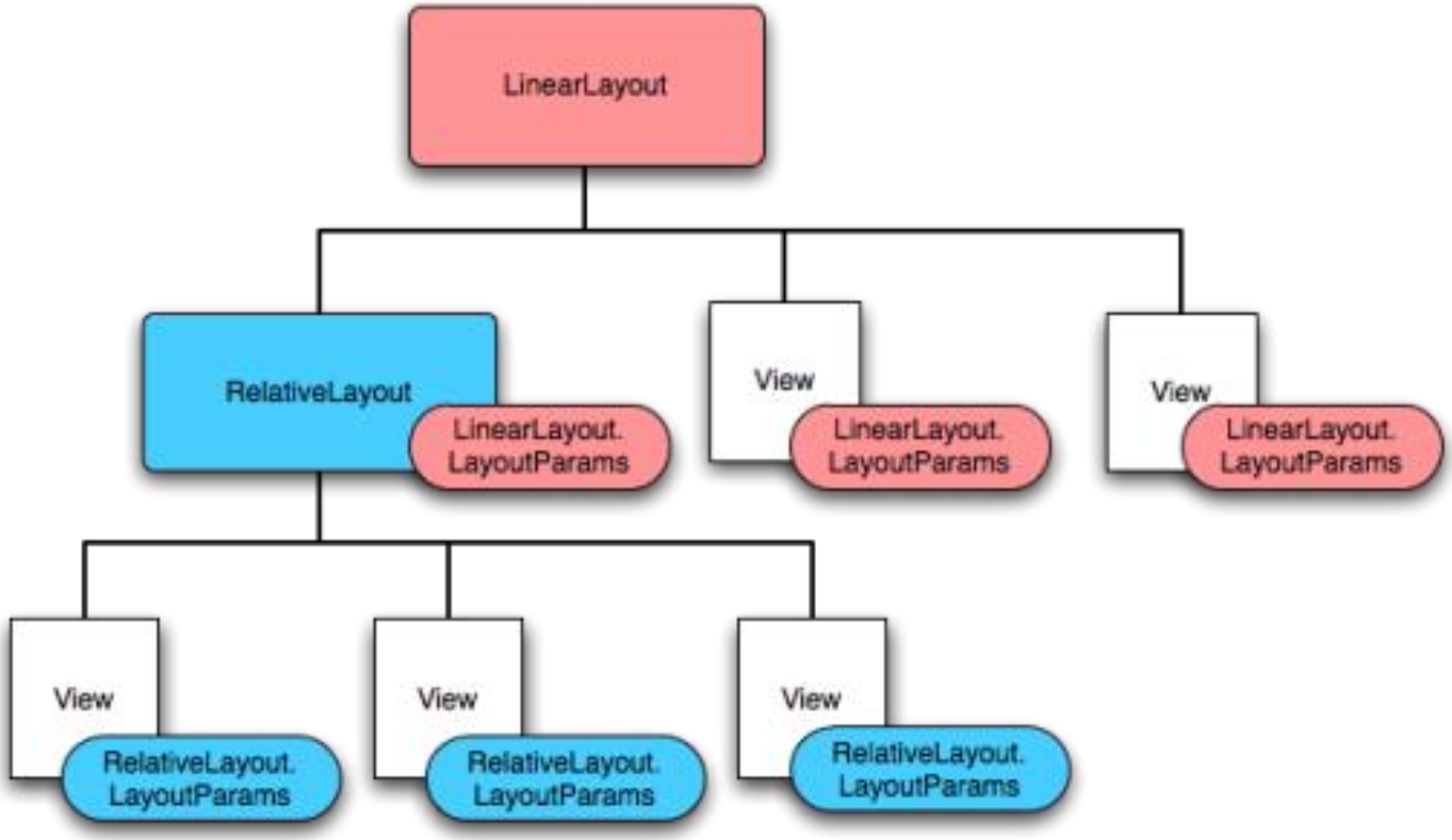
- **android:id** es el nombre de la propiedad
- @ indica que el elemento debe de ser parseado por el compilador.
- + Indica que es un nuevo recurso y que debe de ser agregado a la clase R
- id indica el tipo de recurso y por lo tanto sera agregado en R.id
- Lo que sigue a la / será el nombre con cual podremos referenciar a ese view, asi: R.id.my_button

# Parámetros de layout

Los parámetros de layout son elementos en los cuales definimos como los Views deben de colocarse y dimensionarse dentro del ViewGroup en el cual residen.

- Se encuentran en la sección “Misc” del cuadro de parámetros. Tambien son identificados por el prefijo “layout_”
- Cada subclase que hereda a ViewGroup posee una implementación de ViewGroup.LayoutParams (Clase Genérica). Cada ViewGroup define como interpretar estos parametros

# Parámetros de layout



# Parámetros de layout

Todas las implementaciones de ViewGroup.LayoutParams poseen 2 atributos obligatorios: width (ancho) y height(alto)

- El width y el height pueden ser asignados con valores estáticos(pixeles), aunque no es recomendado para aplicaciones que están destinadas para soportar múltiples resoluciones de pantallas.
- La mejor opción es usar unidades relativas como dp(density-independent pixel units, 160dp es equivalente a una pulgada en cualquier pantalla) o valores calculados por el sistema operativo.

# Parámetros de layout

- Para que Android calcule el tamaño adecuado para un elemento usaremos básicamente 2 modificadores:
  - wrap_content: Especificamos que el tamaño del elemento, es el necesario para mostrar todo su contenido.
  - fill_parent: Especificamos que el tamaño del elemento debe de ser igual que el tamaño de su padre(En Android 2.2 este modificador es llamado *match_parent)*

# Posición y Tamaño de Views

- La posición de un View esta definido por un par de valores que definen el Top y Left. Este valor es devuelto en pixeles(cuando es consultado programáticamente) y corresponde a la posición relativa con respecto a su padre.
- El tamaño de los views esta definido por 2 pares de medidas
  - measured width y measured height: Que tamaño la vista desea tener dentro del padre
  - width and height: El tamaño real con el que fueron dibujados los views