Seminario de Lenguajes
Android
Práctica 1

1) Abrir Android Studio y crear un nuevo proyecto con una Actividad vacía (Empty Activity)

2) Abrir el Android Virtual Device Manager y crear como dispositivo virtual un Galaxy Nexus con el nombre Nexus Seminario Android

3) Probar la aplicación creada en el ejercicio 1 en el emulador

4) Describir qué representa una Activity

5) Abrir el archivo AndroidManifest.xml. ¿Por qué MainActivity tiene un intent-filter con action MAIN y category LAUNCHER?

6) Crear 2 actividades (NuevaActivity1, NuevaActivity2) y ver qué se modificó en el archivo AndroidManifest.xml

7) Pegar el siguiente código en res/layout/activity_main.xml

```
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

        <Button
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Click"
            android:onClick="onBtnClick"/>
</RelativeLayout>
```

8) Añadir el siguiente código en la clase MainActivity.java, probar en el emulador y analizar el resultado

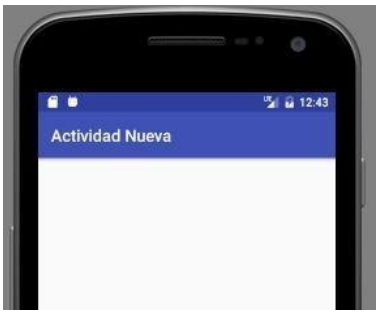
```
fun onBtnClick(view: View?) {
  val i = Intent(this, NuevaActivity2::class.java)
  startActivity(i)
}
```

* Se deben importar estos paquetes: *android.view.View*, *android.content.Intent*

9) ¿Qué significa *this* en el código del ejercicio anterior?

10) ¿Lo que utilizamos fue un intent implícito o explícito? ¿Cual es la diferencia entre ambos?

11) A través del AndroidManifest modificar el nombre que se muestra al usuario para la actividad 2 para que al hacer click se muestre con el texto *Actividad Nueva*



12) Modificar el comportamiento de onBtnClick para que a través de un intent abra una página web

13) En el método onCreate de la actividad nueva añadir la siguiente línea.

Log.*d*(**"APP_DE_PRUEBA"**, **"Este es mi mensaje de debug"**);

Correr la aplicación en modo debug y revisar la consola de Debug luego de abrir la actividad 2. ¿Qué ocurrió? ¿Cuál es su utilidad?

14) Al hacer click en el botón se debe pasar a la actividad 2 un texto como parámetro. Cuando la actividad 2 se muestra se debe imprimir por consola el texto recibido como parámetro.

15) Describa cuales son los estados por los que puede pasar una actividad

16) Describa cuales son los eventos generados a partir de un cambio de estado de una actividad

17) Crear una nueva aplicación en la que se imprima por la consola los cambios de estados de una actividad utilizando lo investigado en el punto 15 y 16

18) Genere una nueva aplicación con una actividad vacía en Android Studio. Edite el archivo AndroidManifest.xml y elimine las siguientes líneas:

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

a) ¿Qué error se produce en el entorno de desarrollo?. ¿Cuál es el motivo del error?
b) Vuelva a colocar el código eliminado para que la aplicación funcione correctamente
c) Agregue un TextView a la Activity generada con el valor "Español" y un botón con el valor 'Cambiar idioma. Al hacer click en el botón el textview debe cambiar su texto a "Inglés". Si se presiona nuevamente sobre el botón, el textview debe contener el valor "Español" nuevamente.

19) Crear una nueva aplicación con 2 actividades como se ven en la figura.

Al hacer click en "Realizar operación" se debe abrir la segunda actividad. Al hacer click en los botones "Incrementar" o "Decrementar" la actividad debe cerrarse y aplicar la operación correspondiente sobre el TextView. Si se presiona "Cancelar" solo debe cerrarse la segunda actividad sin realizar cambios sobre el TextView

A continuación se detallan los layouts de las activities para simplificar el ejercicio (Notar el uso de LinearLayout):

Activity1:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">


        <TextView android:id="@+id/txtContador"
                  android:layout_width="match_parent"
                  android:layout_height="wrap_content"
                  android:text="0"
                  android:textAlignment="center"
                  android:textSize="34sp"
            />

        <Button android:id="@+id/btnRealizarOperacion"
                  android:layout_width="match_parent"
                  android:layout_height="wrap_content"
                  android:text="Realizar operación"
             />
</LinearLayout>
```

Activity2:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="horizontal">

        <Button android:id="@+id/btnIncrementar"
                  android:layout_width="wrap_content"
                  android:layout_height="wrap_content"
                  android:text="Incrementar"
            />

        <Button android:id="@+id/btnDecrementar"
                  android:layout_width="wrap_content"
                  android:layout_height="wrap_content"
                  android:text="Decrementar"
            />

        <Button android:id="@+id/btnCancelar"
                  android:layout_width="wrap_content"
                  android:layout_height="wrap_content"
                  android:text="Cancelar"
            />

</LinearLayout>
```

20) Agregar un control más al ejercicio anterior para que no pueda decrementarse si el valor es 0. Para ello al retornar a la actividad 1 verificar si el valor es 0 y desplegar un mensaje Toast informando el error.

21) Modifique el ejercicio anterior para que la segunda activity (operaciones) de abra con un intent implícito.

22) ¿Qué sucede en el ejercicio anterior si se modifica la orientación del dispositivo (horizontal/vertical)?
    a) Solucione el problema mediante saveInstanceState / restoreInstanceState.

23) Generar una actividad con nombre LifeCycleActivity y pruebe el siguiente código:

```
override fun onDestroy() {
    super.onDestroy()
    val i = Intent(this, LifeCycleActivity::class.java)
    this.startActivity(i)
}
```

    Ejecute la aplicación:
    a) Intente destruir la actividad mediante el botón "Atrás" del dispositivo.
    b) Intente destruir la actividad mediante el botón de intercambio de tareas. (Botón central del Nexus S)

24) En el ejercicio anterior, agregue a la actividad un TextView con id "texto".

    Agregue al método onDestroy el siguiente código:

```
override fun onDestroy() {
    super.onDestroy()
    (findViewById<View>(R.id.texto) as TextView).text = "HOLA MUNDO!"
    val i = Intent(this, LifeCycleActivity::class.java)
    this.startActivity(i)
}
```

    Intente destruir la actividad mediante el botón "Atrás" del dispositivo.
    a) ¿Se ve el mensaje "HOLA MUNDO" en la componente TextView?. ¿Por qué?
    b) ¿Se puede resolver mediante saveInstanceState / restoreInstanceState?
    c) ¿Qué sucede con la instancia de LifeCycleActivity?

25) Crear una nueva aplicación que tenga 2 actividades. En la actividad 1 reemplazar el código del archivo *activity_main.xml* por el presentado a continuación.

```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">
        <!-- Aquí van los botones →
</LinearLayout>
```

La primera actividad debe tener 2 botones. El primero debe tener el texto ¿Qué hora es? y al hacer click debe imprimir en la consola de Debug la hora actual.

El segundo debe contener el texto ¿Qué día es? y al hacer click pasar como parámetro el día de hoy a la 2da actividad. Al abrirse la segunda actividad debe imprimir el valor recibido.

** Investigar la clase SimpleDateFormat para convertir la fecha a String en un formato específico*

Seminario de Lenguajes Android

Práctica 2

1. Crear un nuevo proyecto en Android Studio con una Actividad vacía y modificar el contenido del archivo xml de la actividad creada (en res/layouts), colocar el siguiente código en él y probar en el dispositivo o emulador:

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent">

  <TextView
     android:layout_width="200dp"
     android:layout_height="200dp"
     android:text="Hola Mundo" />
</RelativeLayout>
```

2. Investigue la utilidad de los atributos "android:layout_width" y "android:layout_height".
    a. ¿Qué sucede si se omite alguno de los dos en el elemento RelativeLayout?
    b. ¿Qué sucede si en lugar de "match_parent" se establece como valor "wrap_content"?

3. Añadir un color de fondo al TextView y modificar el color de las letras.

4. Modificar el ancho del TextView para que siempre se adapte al ancho del contenedor. Luego centrar el texto horizontalmente.

5. Cambiar el tamaño de la tipografía (utilizar la unidad de medida sp)

6. Crear un TextView debajo del *Hola Mundo* con el texto *Aquí Estoy*. ¿Qué ocurrió? ¿Por qué?

7. Reemplazar en el código creado *RelativeLayout* por un *LinearLayout* de la siguiente manera y ver el resultado. ¿Para qué sirve especificar la orientación del *LinearLayout*?

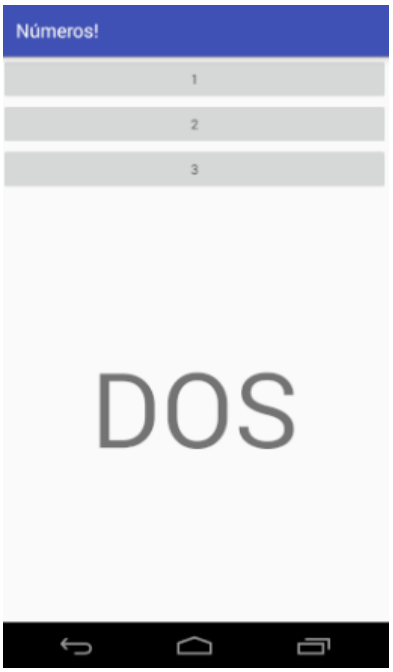
```
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:orientation="vertical">
```

8. Resuelva el problema visual que surgió en el punto 6 sin utilizar *LinearLayout*. (**Nota**: Utilice la propiedad *layout_below*).

9. Genere una disposición de componentes visuales como la siguiente:

10. A partir de la experiencia obtenida con el uso de Layouts.
    a. ¿Por qué cree que son importantes los Layouts?
    b. ¿Cuál hubiera sido la problemática de definir la posición/tamaño de las componentes visuales a través de coordenadas absolutas?
    c. ¿Esta problemática solo existe en dispositivos móviles?

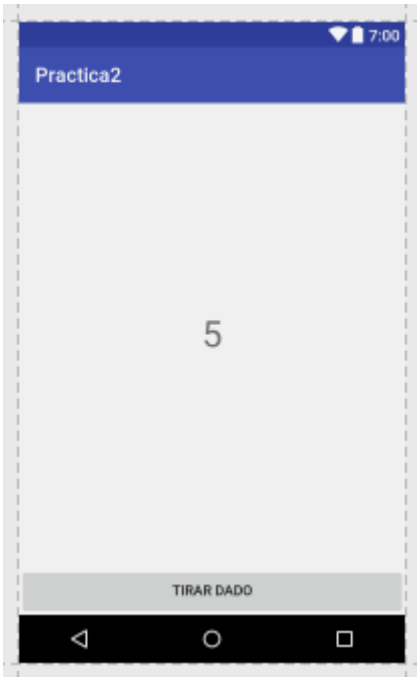
11. Implemente una aplicación con un layout como el siguiente:



Cuando se haga click sobre uno de los botones, la aplicación deberá mostrar en pantalla de forma centrada y con una tipografía más grande la representación en letras mayúsculas del número sobre el cual se hizo click.
El ancho de los botones deberá adaptarse al ancho de la pantalla.
El espacio en donde se visualiza el texto deberá adaptarse al espacio disponible y el texto se mostrará centrado horizontal y verticalmente.

a. Implemente utilizando un manejador de click para cada botón.
b. Implemente utilizando un único manejador de click para todos los botones.

12. Implemente una aplicación con un layout como el siguiente:



Cada vez que se hace click en el botón "Tirar Dado", la aplicación deberá generar aleatoriamente un valor entre 1 y 6 mostrándolo en pantalla.

Para la generación de números aleatorios investigue el uso de la clase java.util.Random

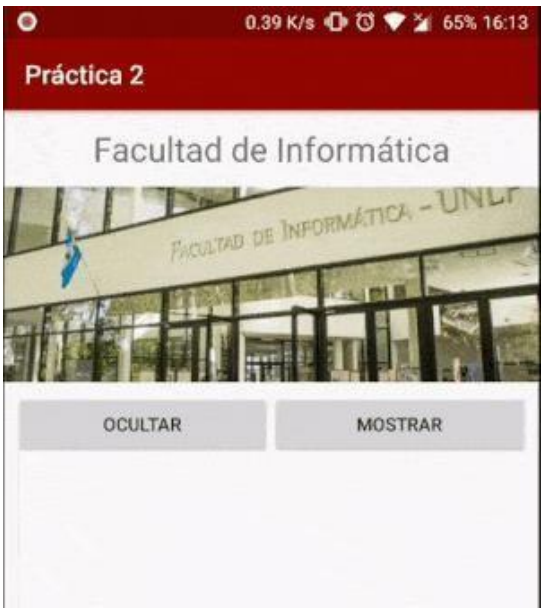
13. Usando un layout similar al desarrollado en el ejercicio 12, agregar una imagen usando el componente **<ImageView>**.

La imagen deberá incluirse en el directorio
`/proyecto/app/src/main/res/drawable/`

Nota: puede usar el siguiente código:

```
<ImageView
        android:src="@drawable/imagen"
        android:layout_width="match_parent"
        android:layout_height="200dp"
        android:scaleType="center" />
```

a. ¿Qué sucede al cambiar el valor del atributo scaleType por "centerCrop"? ¿Y "fitXY"?
b. Al hacer click sobre los botones , deberá ocultarse o mostrarse la imagen usando el método setVisiblity(View.GONE) y setVisibility(View.VISIBLE) sobre la imagen.



¿Qué cambia si en vez de ocultar la imagen usando View.GONE, usamos View.INVISIBLE?

c. Agregar el atributo android:animateLayoutChanges="true" en el LinearLayout que contiene a los elementos. ¿Qué diferencia hay al cambiar la visibilidad de la imagen?

Seminario de Lenguajes Android

Práctica 3

1. Modificar el ejercicio 12 de la práctica 2 para que, al girar el dispositivo, se mantenga el número mostrado en pantalla. Utilice los métodos `onSaveInstanceState/onRestoreInstanceState`.

2. Implemente la siguiente Activity con cuatro imágenes y un título, usando un ScrollView

3. Modificar el ejercicio anterior para que muestre una única imagen. Cada vez que se clickea el botón “Siguiente”, se deberá reemplazar la imagen. Investigar el uso del mensaje `setImageResource` de la clase ImageView.

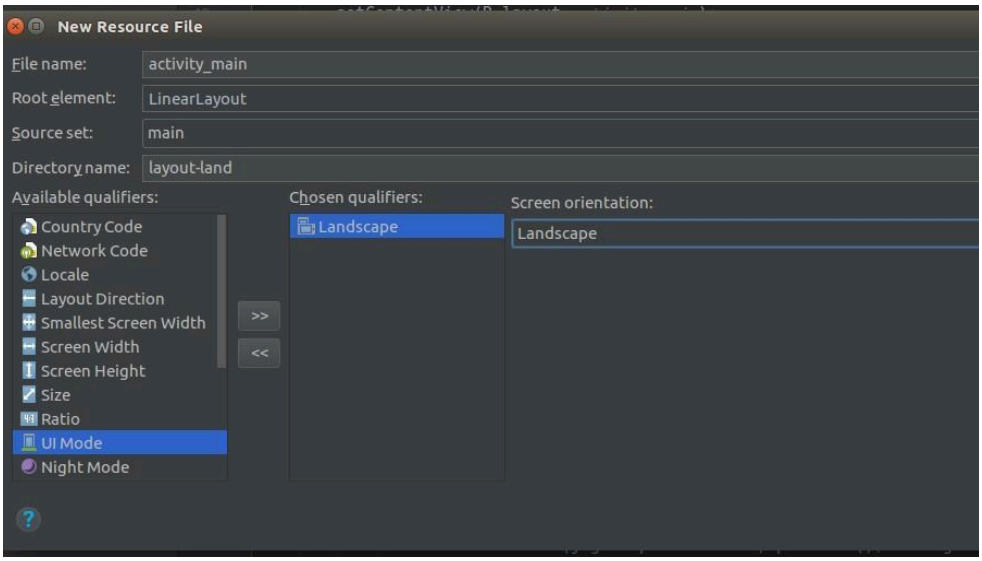
   Al girar la pantalla, no debe cambiarse la imagen y el layout debe visualizarse correctamente.

   Nota: Puede almacenar las referencias a las imágenes en un arreglo de la siguiente manera:


```
private int[] imagenes = {R.drawable.portada, R.drawable.interior, R.drawable.foto3};
```

4. Modifique el ejercicio anterior de manera que al girar la pantalla, se modifique el layout de la siguiente forma.

Implementar la grilla de botones usando GridLayout y un XML diferenciado según la orientación.
Nota: para incluir un layout dependiente de la orientación, desde Android Studio, se debe hacer click derecho en el directorio *layout* → *New* → *Layout resource file*. Deberá tener el mismo nombre de archivo que el layout que usamos para la versión vertical. Luego elegimos "*Orientation*" en el cuadro de "*Available qualifiers*", y luego la orientación *Landscape*:



Por defecto, cuando la aplicación esté en orientación *Portrait*, usará el layout que

habíamos definido originalmente, y ahora la orientación *Landscape* usará este nuevo layout.

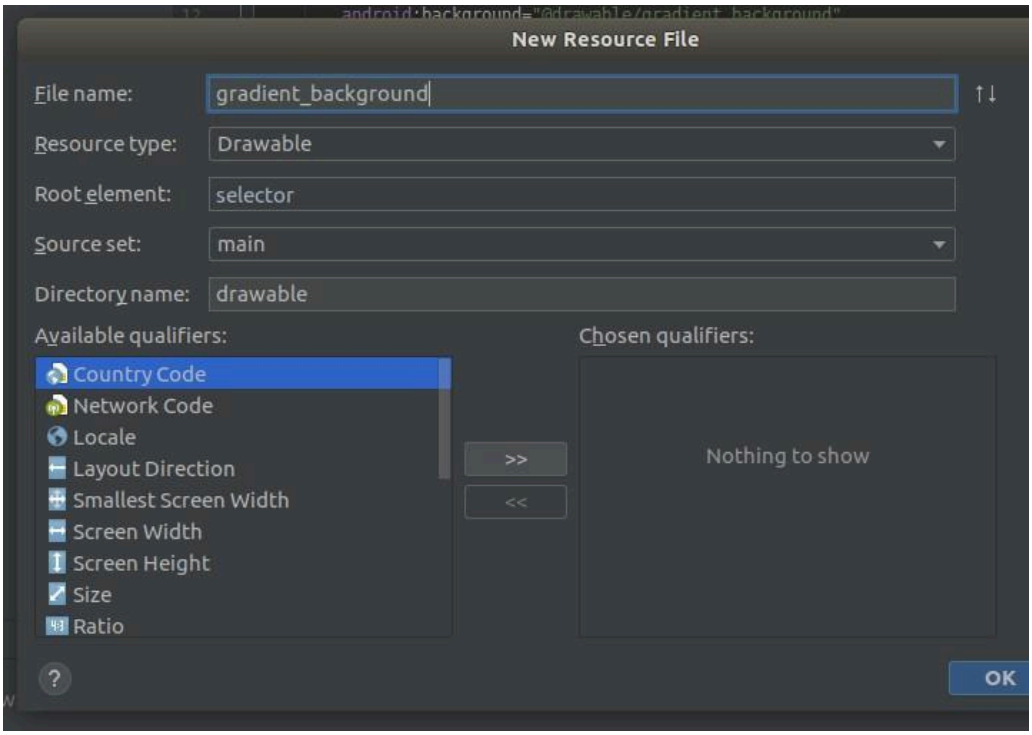
5. Investigue sobre los recursos de elementos de diseño de Android. (Recursos drawable)
    a. ¿Para qué sirven?
    b. ¿Qué tipos de elementos de diseño están disponibles?
    c. ¿Qué tipo de elemento de diseño utilizaría para mostrar una imagen?
    d. ¿Qué tipo de elemento de diseño utilizaría para mostrar un rectángulo con colores?
    e. ¿Qué tipo de elemento de diseño utilizaría para mostrar los estados “presionado” y “suelto” de un botón?

6. Utilice un recurso drawable para definir el fondo de un TextView que se visualice de la siguiente manera:

Para ello deberá:

a. Generar un recurso drawable denominado a modo de ejemplo: *gradient_background*
Click derecho -> New > Android Resource File

Notar que Resource type debe ser: Drawable



b. En el archivo xml generado deberá definir un diseño gradiente vertical lineal entre dos colores. Al mismo tiempo debe definir un borde de 3dp de ancho con un color diferente al del gradiente. Deberá utilizar un Shape. Puede obtener información sobre la sintaxis a utilizar en:

https://developer.android.com/guide/topics/resources/drawable-resource?hl=es-419#Shape

c. Una vez definido el recurso drawable, puede asignárselo a un TextView simplemente estableciendo el atributo android:background de la siguiente manera:

```
<TextView android:background="@drawable/gradient_background"
```

7. Genere un Botón que utilice como background el gradiente definido en el punto anterior. Cuando el usuario presiona el botón, deberá cambiar el gradiente a un color diferente. Para ello, genere un nuevo recurso similar al del punto anterior pero con colores diferentes para el gradiente con nombre: gradient_background_pressed
A continuación defina un nuevo recurso drawable que defina los dos estados del botón y que haga referencia a los drawables previamente definidos. Deberá utilizar un StateList. Puede obtener información de la sintaxis a utilizar en: https://developer.android.com/guide/topics/resources/drawable-resource?hl=es-419#StateList

8. Intente utilizar el drawable StateList definido en el ejercicio anterior en otra componente visual como un ImageView o un TextView. ¿Funciona el cambio de estado al presionar sobre la componente?.

   Establezca el atributo “clickable” de la componente en true y vuelva a intentarlo.