AQUÍ VAMOS A VER:

1. **Imprimir en consola mediante Log.d**
2. **Añadir un botón a una actividad**
3. **Añadir un campo de Texto**
4. **Añadir un TextView**
5. **Añadir librería ButterKnife**
6. **Crear un SnackBar**
7. **Añadir una imagen (y que se oculte al pulsar el botón**
8. **Añadir sonido a un botón**

BOLETÍN DE REPASO

Boletín 1

Problema 1

- *Crear una App Android con una actividad*

Creamos un nuevo proyecto en Android Studio

- *Al iniciarse esa actividad debe imprimir por la consola (mediante Log) el mensaje "Hola Mundo!"*

Para imprimir por consola, utilizaremos el código Log.d (). El Log, requiere dos cosas, una etiqueta, donde se mostrará el problema, y una cadena de texto que será lo que queremos mostrar:

```
Log.d("Este es el problema 1","Hola mundo" );
```

Problema 2

- *Añadir un botón a la actividad*

Las vistas se añaden en el XML de la actividad,(/res/layout, para colocar un simple botón utilizaremos la etiqueta <Button>

Esta vista <Button> tendrá siempre un width (ancho) y un height (alto). Aparte añadiremos siempre un id. Puede llevar muchas más propiedades, texto, color, background…

```
<Button
    android:id="@+id/button"
    android:text="BOTON"
    android:layout_width="match_parent"
    android:layout_below="@id/activity_main___edit_text"
```

```
    android:layout_height="100dp" />
```

- *Al pulsar el botón se debe imprimir el mensaje "Botón pulsado" por la consola (mediante Log)*

Ya en el código, le daremos un nombre a la vistas creadas en el XML, como hemos creado un botón, le daremos un nombre:

```
Button button;
```

Después utilizaremos ese nombre para asignarle su id correspondiente del XML, utilizando el findViewID

```
button = (Button) findViewById(R.id.button);
```

Y por último, al ser un botón, le daremos un OnClickListener, para que realice una acción cada vez que es pulsado. Pondremos el nombre del botón junto a OnClickListener y entre paréntesis, new V y ya se nos rellenará el resto del código automáticamente, Simplemente para que muestre el Hola Mundo en la consola, metemos el código del Log.d anterior, dentro del método del OnClick

```
button.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Log.d("PROBLEMA 2", "El botón ha sido pulsado");
    }
});
```

- *El mensaje anterior "Hola Mundo!" se debe seguir imprimiendo, no borrarlo*

Hecho. =)

Problema 3

- *Añadir un campo de edición de texto a la actividad, donde poder escribir*

El campo de texto lo añadiremos en el XML con un EditText que sirve para recibir lo que el usuario escriba, aquí le he incorporado un **text:hint** que es para que en la caja de texto aparezca una frase que el usuario al pinchar, se borrará.

```
<EditText
    android:id="@+id/activity_main___edit_text"
    android:layout_width="match_parent"
    android:layout_height="200dp"
    android:layout_below="@id/activity_main___text_view"
    android:hint="Escriba algo aquí"/>
```

- *Al pulsar el botón, se debe imprimir por la consola lo que tengamos escrito en el campo de texto*

Repetimos la acción anterior, declaramos el editText poniéndole un nombre:

```
final EditText editText;
```

Y le asignaremos el id del XML con findViewID

```
editText = (EditText) findViewById(R.id.activity_main___edit_text);
```

A continuación, y ya dentro del OnClick, le asignaremos un tipo de valor al editText para evitar errores, en este caso será una cadena de texto.

```
String addText;
```

También le asignaremos un nombre para utilizarlo a continuación.

```
addText = editText.getText().toString();
Log.d(addText, "PROBLEMA 3");
```

Aquí vemos cómo hemos utilizado el nombre dado anteriormente al String que se convierte en la variable que da como resultado el EditText. GetText significa que obtiene (get, significa obtener) el texto introducido por el usuario y toString que es la cadena de texto.

Problema 4

- *Añadir a la actividad una etiqueta TextView*

Volvemos al XML para introducir una nueva etiqueta, en este caso un TextView.

```
<TextView
    android:id="@+id/activity_main___text_view"
    android:layout_width="match_parent"
    android:layout_height="200dp" />
```

- *Modificar el Problema 3 y al pulsar el botón, se debe imprimir dentro de la etiqueta lo que tengamos escrito en el campo de texto*

Le damos de nuevo un nombre al TextView, y le declaramos su ID con findViewID

```
final TextView textview;
```

```
textview = (TextView) findViewById(R.id.activity_main___text_view);
```

Al TextView le asignaremos su propiedad .setText (to set, significa poner) que pondrá lo escrito por el usuario en el cajón de texto, y lo imprimirá.

```
textView.setText(addText);
```

Problema 5

- *Añadir la librería ButterKnife al proyecto*

Para añadir la librería, tenemos que añadir las siguientes líneas en el gradle de nuestra aplicación. En las *dependencies*:

```
compile 'com.jakewharton:butterknife:8.5.1'
annotationProcessor 'com.jakewharton:butterknife-compiler:8.5.1'
```

Y sincronizamos el gradle dándole a Sync Now, tardará unos segundos en añadir la librería.

- *capturar las referencias a todos los controles en la actividad por anotaciones @BindView de ButterKnife*

Para empezar a utilizar la librería ButterKnife, en el OnCreate debemos añadir esta línea de código:

```
ButterKnife.bind(this);
```

Una vez hecho esto, comenzaremos a capturar todas las referencias de nuestro XML poniendo @BindView, la ID del XML y el nombre que le asignamos anteriormente.

Al realizar este proceso, se suprime tanto la asignación del nombre, como su correspondiente findViewID

Quedaría tal que así:

```
//Declaración de vista con ButterKnife
@BindView(R.id.button)Button button;
@BindView(R.id.activity_main___edit_text)EditText editText;
@BindView(R.id.activity_main___text_view)TextView textView;
```

Problema 6

- *Comprobar al pulsar el botón si el texto escrito es ""*

Comprobado y funcionando. =)

- *Si lo es, imprimir un mensaje "Texto corto" mediante un SnackBar*

Turno de colocar un **SnackBar:**
Iremos nuevamente al gradle y añadiremos esta librería:

```
compile 'com.android.support:design:25.1.0'
```

Cuando ya esté añadida la línea de código, en nuestra actividad tendremos que llamarla junto a *make* (make significa “crear”)
El Snackbar requiere de 3 características, una vista, (he probado tanto los nombres del EditText como el TextView y funcionan), un texto, que será el que se muestre, y una duración. Acabará en un .show(); para mostrar el Snackbar.

```
Snackbar.make(editText,"texto cortico",Snackbar.LENGTH_LONG
).show();
```

Problema 7

- *Añadir una imagen al proyecto*

Vuelta al XML añadimos una imagen con la etiqueta <ImageView>, donde como todas las demás, llevará un ID, un ancho y un alto. Le he colocado la propiedad
`android:src="@android:drawable/star_big_off"`
para coger de la galería de android una imagen que haga de referencia.
El código queda de la siguiente manera:

```
<ImageView
    android:id="@+id/activity_main___image_view"
    android:layout_width="match_parent"
    android:layout_height="100dp"
    android:src="@android:drawable/star_big_off"/>
```

- *Mostrar la imagen en la actividad*

Volvemos a utilizar ButtleKnife para pillar la referencia del XML

```
@BindView(R.id.activity_main___image_view)ImageView imageView;
```

- *al pulsar el botón la imagen se debe ocultar (cambiar su visibility).*

En el onClick pillamos el nombre que le hemos dado a su referencia, y le añadimos una de sus propiedades, “visibility”

```
imageView.setVisibility(View.GONE);
```

Entre muchas otras propiedades, la imagen tiene su “visibilidad”, que destacan 3: (VISIBLE > Se mantendrá siempre a la vista)
(INVISIBLE > Al pulsar el botón desaparecerá la imagen, dejando su hueco)
(GONE > Al pulsar el botón desaparecerá la imagen, y su hueco)

Problema 8

- *Añadir un fichero con un sonido al proyecto*

Nos bajamos un archivo de audio de internet, y lo colocamos en la carpeta raw, si no existe, la creamos, debe de estar dentro del paquete <res>

- *Hacer que suene al pulsar el botón (además de imprimir lo que tengamos en pantalla)*

Para hacer que suene el sonido al pulsar el botón, en el MainActivity creamos el objeto MediaPlayer y le asignamos el nombre, (instanciamos)

```
MediaPlayer mediaPlayer;
```

Una vez hecho, en el OnCreate le asignamos el sonido que pusimos en la carpeta <raw>.
También llevará la propiedad .create, para como dice, crear el MediaPlayer con ese sonido:

```
mediaPlayer = MediaPlayer.create(this, R.raw.sound_button);
```

Y ya por último, para que se reproduzca el sonido al hacer click en el botón, añadimos en el OnClickListener la variable y su propiedad start();

```
mediaPlayer.start();
```

FIN