

## Ejercicios de repaso

*Indicaciones generales:*

- Las apps de estos ejercicios harán uso del **API Level 26**
- Todas las apps constarán de un **Fragment** que se mostrará en **MainActivity**
- Todas las apps deberán conservar su estado tras un cambio estructural
- Las librerías de Java funcionan en Kotlin. Cualquier tarea para manipular listas, fechas, archivos, generar aleatorios, etc se hace igual que en Java

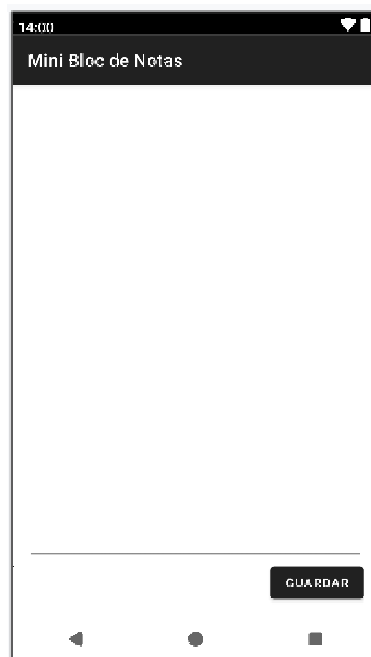
**Ejercicio 1 :** Realiza una app con el diseñador de Android Studio, que tenga estos elementos:

- Dos **Guideline** para alinear los componentes con una separación de **16dp** a los bordes izquierdo y derecho de la pantalla.
- Un **EditText** que permita introducir un número con decimales, alineado a las guías, y con el máximo tamaño posible.
- Un **EditText** que permita introducir un segundo número con decimales, igual que el anterior
- Un **Flow** que contenga cuatro **Button** con los signos de las operaciones de sumar, restar, multiplicar y dividir. Los botones se verán centrados en la pantalla.
- Un **Barrier** que permita a la interfaz crecer correctamente si se aumenta el número de botones
- Un **EditText** que mostrará el resultado, y que será de solo lectura.

Al pulsar cualquiera de los botones de las operaciones, se realizará la operación correspondiente entre los números escritos en los cuadros de texto, y el resultado se mostrará en el último cuadro de texto.

**Ejercicio 2 :** Realiza una app que tenga una toolbar fija, un **EditText** que ocupe toda la pantalla y un **Button** al final, alineado abajo a la derecha. La app será una especie de bloc de notas simplificado, que funcionará de esta forma:

- El usuario podrá escribir un texto cualquiera en el **EditText**
- Al pulsar el botón, se creará (o sobrescribirá si ya existe), en el directorio de la app un archivo llamado **bloc\_notas.txt** cuyo contenido será lo que esté escrito en el cuadro de texto.
- Al iniciarse la app, se cargará el contenido de **bloc\_notas.txt** y se mostrará en el cuadro de texto.



Ayuda:

*Las clases para trabajar con archivos son las mismas que se usan en Java (InputStream, Reader, PrintWriter, BufferedWriter, File, Files, Path, etc). Lo único que tienes que tener en cuenta es que primero debes obtener un objeto **File**, usando el siguiente constructor:*

**+ File(File directorioPadre, String nombreArchivo)**

*El directorio de la app nos lo da el método **context.GetFilesDir()** y se usa como parámetro **directorioPadre** en el método anterior.*

*Una vez obtenido el **File**, puedes usar las demás clases de Java con normalidad.*


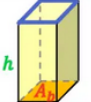




**Ejercicio 3 :** Realiza una app con el diseñador de Android Studio, para jugar al juego de acertar un número. Las instrucciones son las siguientes:

- El juego comienza en el nivel 1, el usuario tiene 10 vidas, ha realizado 0 intentos para acertar el número y tiene 0 puntos
- Se genera un número secreto de forma aleatoria entre 1 y 100.
- La pantalla tendrá un **EditText** para pedir un número al usuario, un **Button** para que el usuario adivine el número y un **TextView** que inicialmente no se ve.
- La pantalla también mostrará el número de nivel, el número de vidas, los intentos usados para acertar el número y los puntos
- Al pulsar el **Button**:
  - Se hará visible el **TextView**
  - Se sumará un intento para acertar el número
  - Se mostrará el mensaje “Te has pasado” o “Te has quedado corto”, según el número introducido por el usuario sea mayor o menor que el número secreto. El color del mensaje se obtendrá calculando el valor absoluto de la diferencia entre el número introducido por el usuario el número secreto, y se mirará el color que corresponde a dicho valor en esta tabla:

Menor de 20	Entre 20 y 40	Entre 40 y 60	Entre 60 y 80	Mayor de 80
Rojo	Naranja	Amarillo	Azul claro	Azul oscuro

- Si el usuario falla, se restará una vida, y si se acaban las vidas se mostrará un **Toast** indicando cuál era el número secreto, y se volverá al nivel 1 con 10 vidas y un nuevo número aleatorio.
- Si el usuario acierta, se mostrará un **Toast** indicando que ha acertado, se sumarán **(nivel\*10) – nº intentos** puntos a su marcador, se incrementará el número de nivel del juego y se elegirá otro número aleatorio.

**Ejercicio 4 :** Tenemos estas figuras geométricas, con su área y su volumen:

Cuerpo geométrico	Forma	Área	Volumen
Cubo		$A = 6 \cdot L^2$	$V = L^3$
Prisma		$A = 2 \cdot A_b + P_b \cdot h$	$V = A_b \cdot h$
Pirámide		$A = A_b + \frac{P_b \cdot ap}{2}$	$V = \frac{1}{3} \cdot A_b \cdot h$
Cilindro		$A = 2 \cdot \pi \cdot r \cdot (r + h)$	$V = \pi \cdot r^2 \cdot h$
Cono		$A = \pi \cdot r \cdot (r + g)$	$V = \frac{1}{3} \cdot \pi \cdot r^2 \cdot h$
Esfera		$A = 4 \cdot \pi \cdot r^2$	$V = \frac{4}{3} \cdot \pi \cdot r^3$

Realiza una app, usando la vista de código fuente xml, que tenga como elemento raíz un **LinearLayout** vertical con los siguientes componentes visibles:

- Un **LinearLayout** horizontal que contenga:
  - Un **Spinner** en el que se mostrarán los nombres de todas las figuras
  - Un **Button** que permitirá seleccionar una figura del spinner

Y fuera del **LinearLayout** horizontal, los siguientes componentes no visibles:

- Varios **EditText** para introducir lado (L), área de la base ( $A_b$ ), perímetro de la base ( $P_b$ ), altura (h), apotema (ap), radio (r) y generatriz (g). Cuando el usuario pulse el botón, se harán visibles solo los cuadros de texto correspondientes a la figura elegida (por ejemplo, si se elige cono, se mostrarán solo el radio, generatriz y la altura).
- Un **ImageView** que cuando el usuario pulse el botón, mostrará la imagen de la figura elegida por el usuario
- Un botón **calcularArea** y **calcularVolumen**, que se harán visibles al pulsar el botón, y que mostrarán en un **Snackbar** el área y volumen de la figura seleccionada

*Sugerencia: Se recomienda la creación de una interfaz **FiguraGeométrica** (en Kotlin es similar a Java) que tenga los métodos comunes a todas las figuras geométricas, y luego realizar una clase que implemente dicha interfaz, para cada figura (si no se hace así, es necesario realizar un montón de bloques **when**, que es muy engorroso).*

**Ejercicio 5**: Realiza una app que muestre en su interior la imagen de las áreas y volúmenes del ejercicio anterior (*archivo **figuras.png** adjunto a los ejercicios*). La imagen se mostrará sin escalar verticalmente, de forma que será necesario subir y bajar la pantalla para poder verla.

Realiza tres versiones del ejercicio:

- a) Con una **toolbar** fija en la pantalla
- b) Con una **scrolling toolbar**
- c) Con una **collapsing toolbar** de 300dp de altura, usando una imagen de fondo