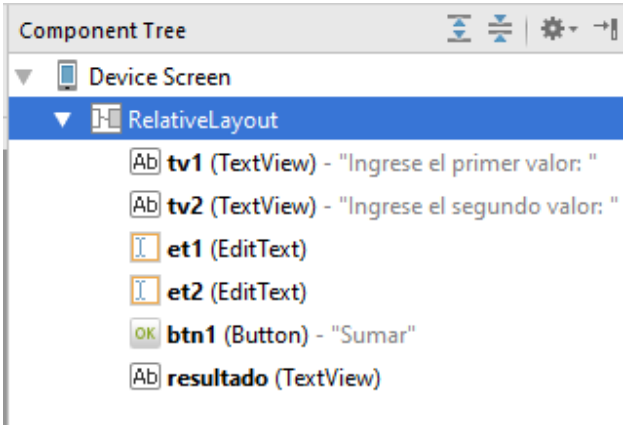
Ejercicio 13: Introducción a la programación de Android Studio

En este decimotercer ejercicio vamos a comenzar a ver algunos conceptos de programación de programación de este lenguaje de programación.

Cuando vamos a programar en este lenguaje debemos tener en cuenta que primero debemos crear el formulario y ponerle nombre a todos los componentes, como por ejemplo en el ejercicio de la suma:

En este ejemplo tenemos en el formulario, la pantalla del móvil declaradas varias estructuras que podemos en ver en modo diseño con sus correspondientes nombres:



Hay que declarar todas las variables que vayamos a usar en el programa

Para poder usar los valores que introducimos en pantalla en este lenguaje debemos almacenarlos en una variable dentro del código fuente, para ello primero declaramos las variables que vamos a necesitar en nuestro código fuente, esto lo hacemos en el MainActivity.java, y en este primer programa lo hicimos de esta manera:

```
EditText n1, n2;
Button  btnSumar;
TextView r;
```

Seguidamente, debemos pasar las variables de tipo texto que entran por pantalla a variables de tipo texto dentro del programa, esto lo hicimos con:

```
n1=(EditText)findViewById(R.id.et1);
n2=(EditText)findViewById(R.id.et2);
r= (EditText)findViewById(R.id.resultado);
```

Esta es una gran diferencia con Visual Basic por ejemplo, que no necesitamos realizarlo. Luego si queremos usar esas variables como números, debemos convertir esas variables de tipo texto en variables enteras, usando la conversión de texto a enteros, como se puede ver:

```
int valor1 = Integer.parseInt(n1.getText().toString());
int valor2 = Integer.parseInt(n2.getText().toString());
int s = valor1+valor2;
```

Como se puede ver, una de las cosas que permite Android es declarar una variable, y a la vez introducirle un valor, una diferencia sobre Visual Basic y Pascal.

Una de las cosas curiosas de este lenguaje de programación es que si queremos poner el resultado de la suma en pantalla, nuestra lógica de otros lenguajes vistos anteriormente, nos diría que escribiésemos este código:

```
r.setText(s);
```

Pero si lo usamos no funcionaria, en el resultado al pulsar el botón nos aparecería el resultado en blanco porque hay que convertir la variable s a texto, y eso se puede hacer fácilmente, añadiendo un espacio en blanco al resultado, una forma sencilla de realizarlo sin usar la conversión de tipos, sería de la siguiente manera:

```
r.setText(s+"");
```

Finalmente una de las cosas especiales de este lenguaje, es el evento click, que como has podido ver en el ejercicio 9 se crea haciendo control y click, como pudiste ver en el video, todas las variables creadas dentro de él sólo serán visibles en ese procedimiento. Creamos el siguiente código para el evento en el programa:

```
public void onClick(View v) {
    int valor1 = Integer.parseInt(n1.getText().toString());
    int valor2 = Integer.parseInt(n2.getText().toString());
    int s = valor1+valor2;
    r.setText(s+"");
}
```

- *Una de las cosas mas importantes de este lenguaje es que* **distingue entre mayúsculas y minúsculas**, *las variables r y R, no son las mismas.*

- *El comienzo y fin se realiza con un* **abre llave y con un cierra llave***(comienzo:{, fin:}), el equivalente en otros lenguajes de programación al begin y end.*

{

:}

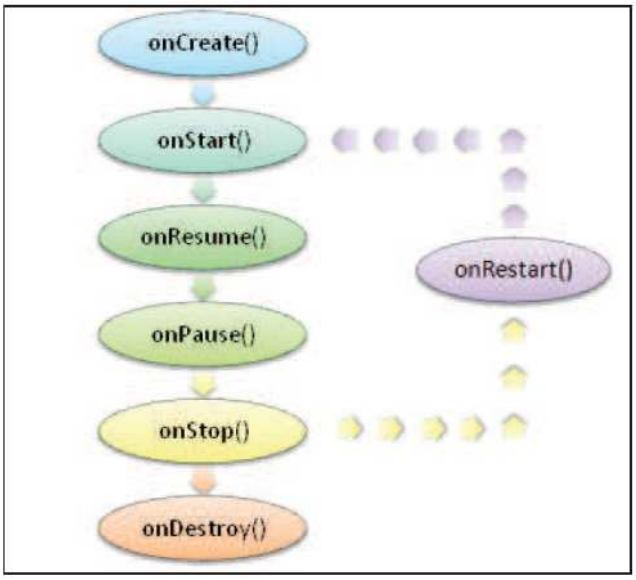
Métodos: *Al gual que en Visual Basic, está orientado a objetos y los* ***métodos*** *más importantes de una actividad:*

- ***OnCreate*** *(Bundle savedInstanceState): es el método que crea la actividad. Recibe un parámetro de tipo Bundle, que contiene el estado anterior de la actividad, para preservar la información que hubiera, en caso de que hubiera sido suspendida, aunque también puede iniciarse con un null si la información anterior no es necesaria o no existe.*
- ***OnRestart():*** *reinicia una actividad tras haber sido parada (si continúa en la pila de tareas, o sea había sido activada anteriormente). Se inicia desde cero.*
- ***Onstart():*** *inmediatamente después de onCreate(Bundle savedInstanceState), o de onRestart() según corresponda. Muestra al usuario la actividad. Si ésta va a estar en un primer plano, el siguiente método debe ser onResume(). Si por el contrario se desarrolla por debajo, el método siguiente será onStop(). Es recomendable llamar al método onRestoreInstanceState() para asegurar la información*
- ***OnResume():*** *establece el inicio de la interactividad entre el usuario y la aplicación. Solo se ejecuta cuando la actividad está en primer plano. Si necesita información previa, el método onRestoreInstanceState() aportará la situación en que estaba la actividad al llamar al onResume(). También puede guardar el estado con onSaveInstanceState().*
- ***OnPause():*** *se ejecuta cuando una actividad va a dejar de estar en primer plano, para dar paso a otra. Guarda la información, para poder restaurar cuando vuelva a estar activa en el método onSaveInstanceState(). Si la actividad vuelve a primer plano, el siguiente método será onResume(). En caso contrario, será onStop().*
- ***OnStop():*** *la actividad pasa a un segundo plano por un largo período. Como ya se ha dicho, el sistema puede liberar el espacio que ocupa, en caso de necesidad, o si la actividad lleva parada mucho tiempo.*
- ***OnDestroy():*** *es el método final de la vida de una actividad. Se llama cuando ésta ya no es necesaria, o cuando se ha llamado al método finish().*

Además de estos ***métodos,*** *cabe destacar dos más, que son de vital importancia:*

- ***OnSavedInstanceState():*** *guarda el estado de una actividad. Es muy útil cuando se va a pausar una actividad para abrir otra.*
- ***OnRestoreInstanceState():*** *restaura los datos guardados en onSavedInstanceState() al reiniciar una actividad.*

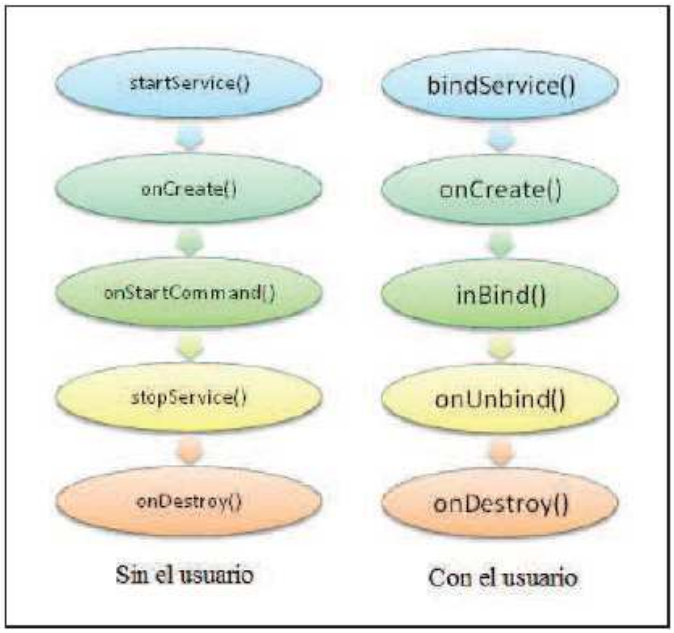
El ciclo de vida de una actividad es:



Servicios: *Los servicios (o service) son tareas no visibles que se ejecutan siempre por debajo, incluso cuando la actividad asociada no se encuentra en primer plano. Tiene un hilo propio (aunque no se pueden ejecutar solo), lo que permite llevar a cabo cualquier tarea, por pesada que sea. No necesita interfaz, a no ser que se pida explícitamente, en cuyo caso la clase Service la exportaría.*
El ciclo de vida de un servicio se inicia con el método onCreate(Bundle), y se libera con el método onDestroy(). Sin embargo, el desarrollo puede llevarse a cabo de dos maneras, dependiendo de cómo se lance:

- *Si se llama al método startService(), esto implicará que el servicio ejecutará todo su ciclo vital. El siguiente método tras onCreate(Bundle) será onStartComand(Intent, int, int). Para terminar el servicio externamente, se usa stopService(), e internamente, stopSelf() ó stopSelfResult(), ambos de la clase Service.*
- *En otro caso, si el servicio se llama con bindService(), el usuario podrá interactuar mediante la interfaz que exporta el servicio, y tras onCreate(Bundle) se ejecutará el método onBind(Intent). En este caso, el servicio se termina llamando al método onUnbind(Intent). También es posible reiniciarlo con el método onRebind(Intent).*

El ciclo de vida de un servicio es:



LAYOUTS EN XML: *Un layout es un recurso con el que puedes describir lo que quieres mostrar por pantalla y cómo lo quieres mostrar. La manera más común de crearlo es a través de un archivo XML (en el directorio res/layout del proyecto), con un formato muy similar a HTML, que sigue este patrón:*

```
public void onCreate(Bundle savedInstance) {
super.onCreate(savedInstance);
setContentView(R.layout.nombre_del_layout);
}
```

Todos los layouts tienen unos parámetros específicos. Los únicos que son comunes a todos son layout_height y layout_width, que sirven para especificar el valor de la altura y anchura del entorno, respectivamente. Aunque se pueden emplear valores numéricos, con "wrap_content", el tamaño se ajusta a las dimensiones del contenido. Con "fill_parent", será tan grande como lo permita su padre o contenedor. Estas 2 opciones son más recomendables que el ajuste manual. También se pueden establecer y consultar los márgenes, bordes y la posición del layout, entre otras opciones, mediante métodos y funciones a los que, en esta ocasión, se llaman desde el código de la

aplicación.
Cada componente tiene su propia variedad de atributos en XML. El atributo "ID" se encarga de distinguirlos del resto, otorgándole un nombre único. Una vez establecido (mediante, por ejemplo, android:id="@+id/nombre"), para referenciarlo desde el código de la aplicación es preciso hacerlo de esta manera:

```
Tipo myTipo = (Tipo) findViewById(R.id.nombre);
```

Existen otros muchos atributos, que varían dependiendo del componente con el que estemos tratando, y con los que podemos establecer el color de fondo, tamaño, gravedad y un sinfín de opciones más.

- *FRAME LAYOUT*
  Es el más simple de todos los existentes. Todos los objetos que se introduzcan se situarán en la esquina superior izquierda, por lo que si hay más de uno, se ocultarán total o parcialmente entre ellos, salvo que los declaremos como transparentes. Por este motivo, su uso ideal es el de mostrar una sola imagen que complete toda la pantalla.
- *LINEAR LAYOUT*
  Se trata del Layout que viene por defecto, y uno de los más sencillos. Los objetos son estructurados horizontal o verticalmente, dependiendo del atributo "orientation" de su archivo XML correspondiente, y siempre en una única fila o columna, con un comportamiento similar al de una pila. Aquí podemos ver un código de ejemplo:

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:orientation="vertical">
<EditText android:id="@+id/NombreEditText"
android:layout_width="fill_parent"
android:layout_height="fill_parent" />
<Button android:id="@+id/NombreButton"
android:layout_width="wrap_content"
android:layout_height="fill_parent" />
</LinearLayout>
```

- *TABLELAYOUT*
  Utilizando esta opción, se consigue una distribución tabular de los elementos de nuestra interfaz. El comportamiento es similar al empleado en HTML: se definen las filas, y dentro de ellas, las columnas. La tabla tendrá tantas columnas como la fila con un mayor número de celdas. En cada casilla, se podrá introducir el objeto deseado (e incluso dejarla vacía). También existe la posibilidad de combinar celdas. Por lo general, la dimensión de cada casilla la determina el elemento contenido en ella. No obstante, este comportamiento puede variarse utilizando diversos atributos.
- *RELATIVE LAYOUT*
  Es la opción que ofrece más posibilidades, y por tanto, la más compleja. Básicamente, empleando este Layout podremos colocar cada elemento en el lugar que deseemos, basándonos en su posición relativa al padre (contenedor) o a otros elementos existentes, pudiendo modificar las distancias entre objetos al antojo del programador.
- *OTROS*
  Existen, además, otros elementos con comportamiento similar a los Layouts que hemos visto hasta ahora. Algunos de ellos proporcionan estructuras invisibles para el usuario (como los aquí descritos hasta ahora) y otros sí que las muestran. Algunos ejemplos son Gallery, GridView, Spinner o ViewFlipper.

EVENTOS DE USUARIO: *Los eventos de usuario sirven para capturar la interacción del usuario con una determinada aplicación. Existen varias maneras de realizarlo.*

- *EVENT LISTENERS*
  Un Event Listener es una interfaz de la clase View a través de la cual se pueden detectar diferentes tipos de pulsación del usuario sobre el dispositivo. Los métodos más comunes que incluye esta interfaz son:
  - ***onClick():*** *este método es llamado cuando el usuario hace una pulsación simple, ya sea por contacto, con teclas de navegación o de cualquier manera posible, sobre un determinado elemento de la interfaz. Una posible implementación, sobre un botón:*

```
private OnClickListener nombreEvento = new OnClickListener() {
public void onClick(View v) {
// hacer lo que se desee
   }
};
Button boton = (Button)findViewById(R.id.corky);
boton.setOnClickListener(nombreEvento);
```

  - ***onKey():*** *este método es llamado si el usuario presiona determinada tecla o botón de su dispositivo mientras el cursor está situado sobre el elemento deseado.*
  - ***onTouch():*** *este método es llamado cuando el usuario toca la pantalla y realiza una presión, se despega o se mueve por ella.*

  Para implementar los métodos aquí vistos y otros existentes, es preciso implementarlos en la Activity correspondiente o definirlos como una nueva clase anónima. Tras esto, deberemos pasar una instancia de la implementación a su respectivo método, con, por ejemplo, setOnClickListener() de OnClickListener.

- *EVENT HANDLERS*
  Si se está creando un componente de la clase View sobre cualquier tipo de Layout, se pueden definir varios métodos por defecto, denominados Event Handlers, que son llamados cuando se produce un evento de cualquier tipo, como puede ser una pulsación sobre la pantalla.

- *TOUCH MODE*
  Determinados dispositivos pueden ser controlados a través de un cursor con sus respectivas teclas de movimiento, tocándoles la pantalla, o ambas. En el primer y último caso, es preciso contar con un selector de componente cuya función sea advertir sobre el elemento con el que se está tratando en cada momento, dependiendo del movimiento y situación del cursor. Sin embargo, en el momento que el usuario "toca" la pantalla, no es necesario resaltar el elemento con el que se va a trabajar. Por ello, en caso de tratar con un dispositivo que permita ambos tipos de navegación a través de su interfaz, existirá un modo llamado "touch mode", que determinará si es necesario resaltar el componente con el que se va a trabajar o no.

- *HANDLING FOCUS*
  Por lo comentado en el punto anterior, el sistema debe encargarse del selector de componente, manejando una rutina, en respuesta a la entrada dada por el usuario. Esto incluye el dónde situar el foco si un elemento es suprimido u ocultado, basándose en un algoritmo que selecciona al componente vecino más cercano.

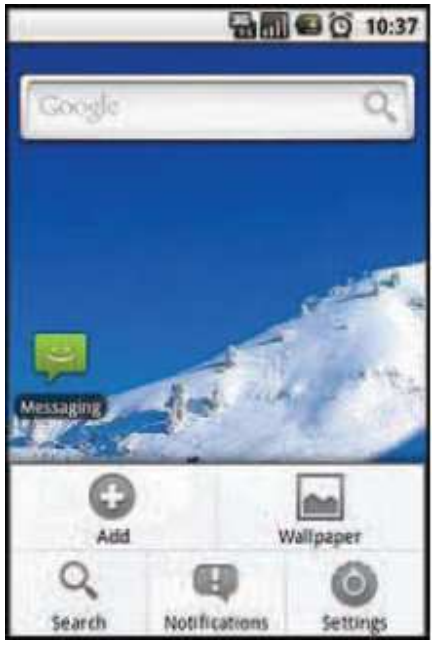
MENUS Y BARRAS DE ACCIONES: *Los menús son la forma más habitual de proporcionar al usuario una serie de acciones a realizar, ya sea sobre una aplicación o sobre las opciones del propio dispositivo. Para crearlo, la mejor opción es definirlo en un archivo XML que irá contenido en el directorio res/menú del proyecto. Los elementos que lo componen son <menú>, que es el contenedor principal del resto de elementos; <ítem>, que representa cada una de las opciones que ofrece el menú, y <group>, que posibilita agrupar los "ítems" del menú a gusto del usuario, y que puede ser visible o no. Por ejemplo:*

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">
<item android:id="@+id/Opcion1" android:title="Opción1"
android:icon="@drawable/miIcono1"></item>
<item android:id="@+id/ Opcion2" android:title="Opción2"
android:icon="@drawable/ miIcono2"></item>
</menu>
```

Hay 3 tipos de menús de aplicación:

- ***Menú principal:*** *es la colección primaria de una actividad, que aparece cuando el usuario acciona el botón "Menú" del dispositivo. Una forma de crearlo es esta:*

```
public boolean onCreateOptionsMenu(Menu menu) {
MenuInflater inflater = getMenuInflater();
inflater.inflate(R.menu.miMenu, menu);
return true;
}
```



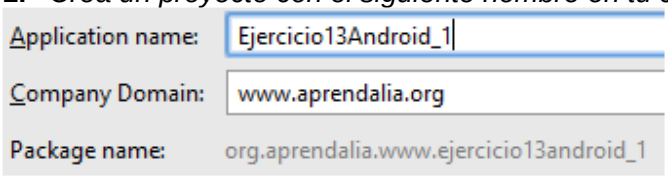
- ***Menú contextual:*** *es una "lista" que surge en la pantalla que aparece cuando el usuario mantiene presionado un elemento determinado. Se implementa como el caso anterior, pero añadiendo un nuevo elemento de tipo <menu> al mismo nivel que los elementos <ítem>.*

-Submenús: *es una lista de opciones que surge cuando el usuario acciona un elemento del menú, que contiene otro menú anidado. Las barras de acciones son barras de título que, además de mostrar algún tipo de información, pueden ejecutar acciones. El aspecto de ésta es acorde con el de la aplicación a la que pertenece. Es importante saber que sólo están disponibles para versiones de Android a partir de la 3.0 (versión 11 del sdk).*

ACTIVIDAD

Sigue los siguientes pasos ahora:

1. *Crea un proyecto con el siguiente nombre en tu carpeta de alumno:*

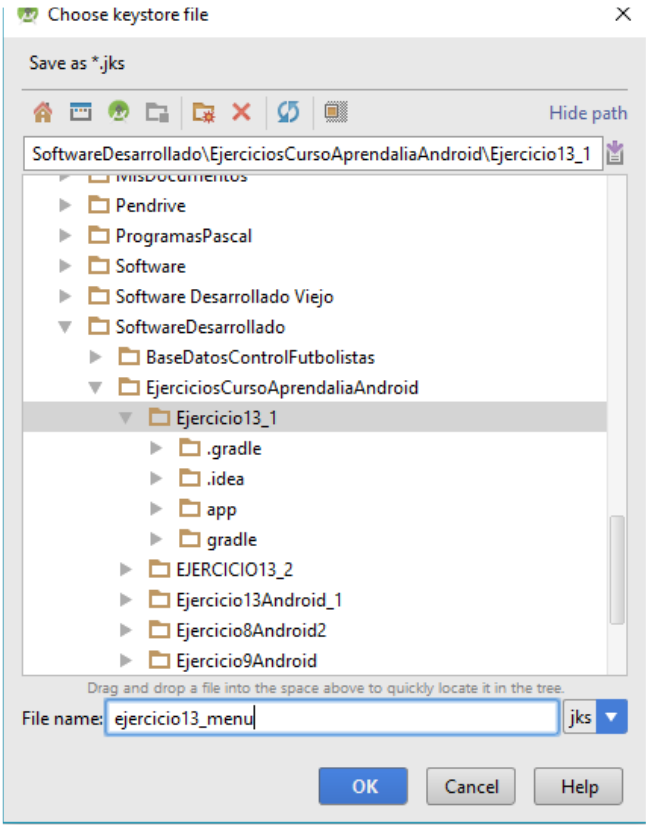


2. *En el modelo de entrada debemos seleccionar:*
    a. *Blank Activity*
3. *En el proyecto debes seleccionar:*
    a. *Plataforma: Teléfono y tablet.*
    b. *Versión: API 16 de Android.*
    c. *Empty activity.*
    d. *Activity Name: MainActivity*

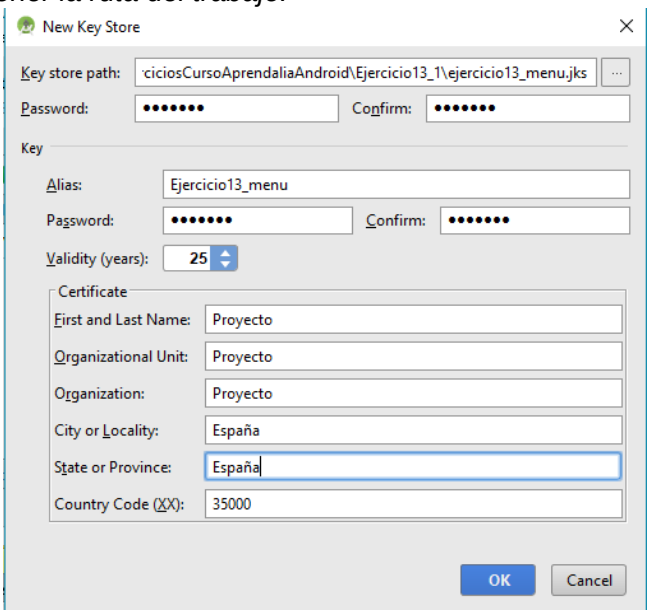
4. *Debes crear el proyecto con* ***blank_activity***
5. *Mira el siguiente video para ver como crear un menú:*

Ver como crear menús

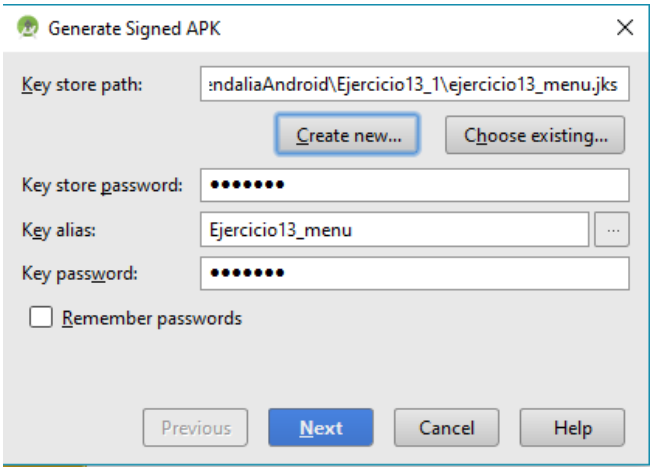
6. *Una vez hayas creado el documento debes generar la aplicación como has visto en el video guardando el fichero generado como se muestra:*



7. *Completa luego la pantalla siguiente, poniendo en contraseña* **colegio** *y en la ruta para guardarlo debes poner la ruta del trabajo.*



8. *Luego completar la siguiente pantalla:*



9. *Muestra el resultado al profesor.*
10. *Guarda el trabajo en tu carpeta de alumno con el nombre* **Ejercicio13Android** *y muéstraselo al profesor.*