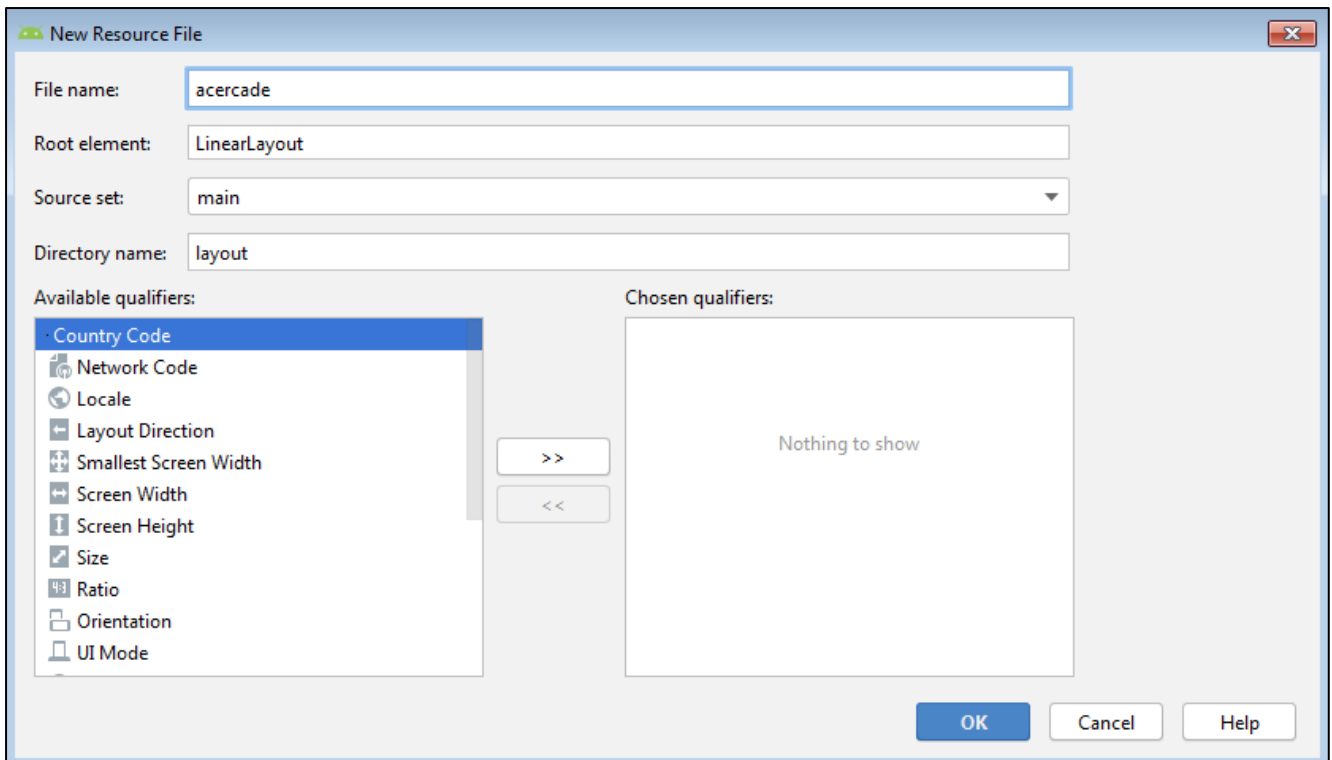
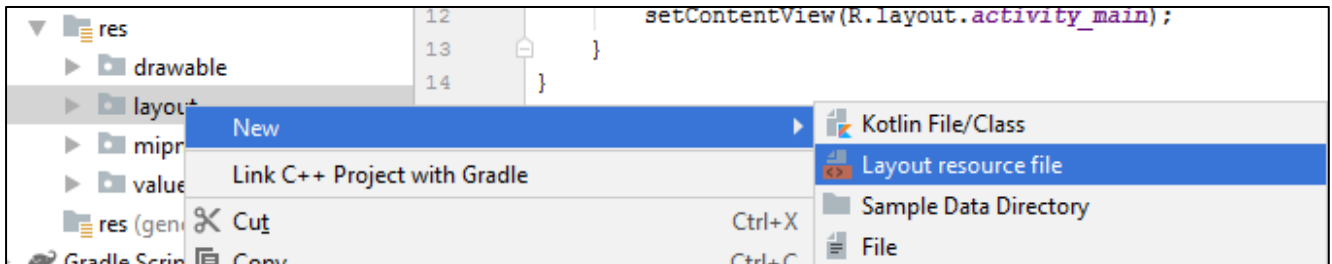


PRACTICA 11: ASTEROIDES

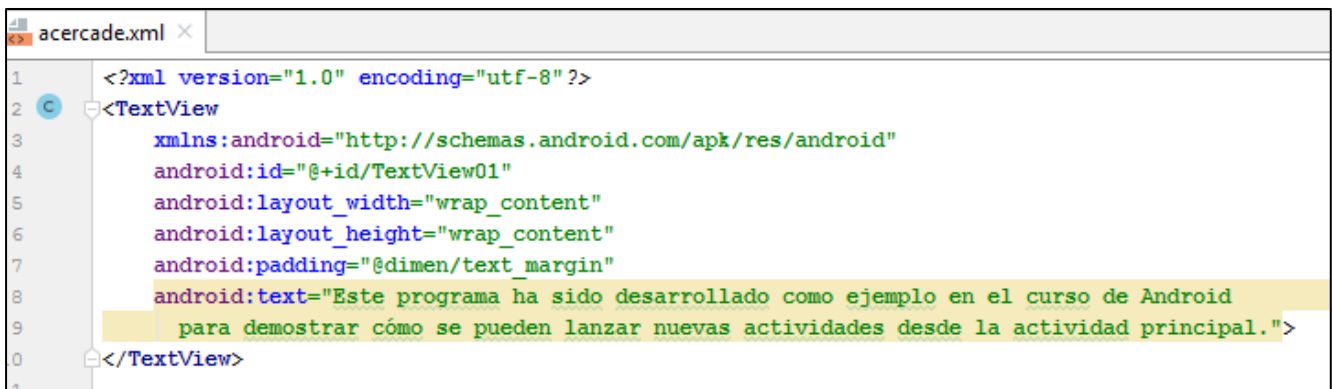
Parte I: Implementación de una caja Acerca de

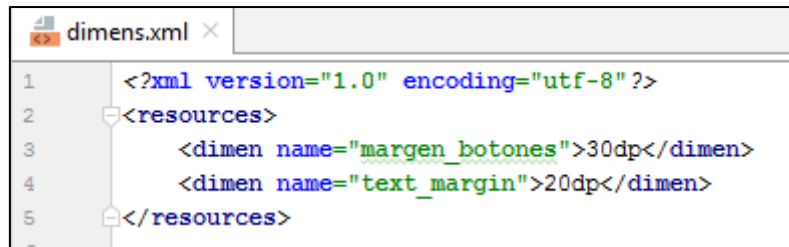
Dentro del proyecto Asteroides, vamos a crear una caja *Acerca de...* y visualizarla cuando se pulse el botón adecuado.

Paso 1. En primer lugar crea el fichero `res/layout/acercade.xml`. Para ello pulsa con el botón derecho sobre el explorador del proyecto en la carpeta `res/layout` y selecciona `New > Layout resource file`. Indica en File name: `acercade`.



Paso 2. Selecciona la lengüeta `Text` y copia el siguiente contenido:



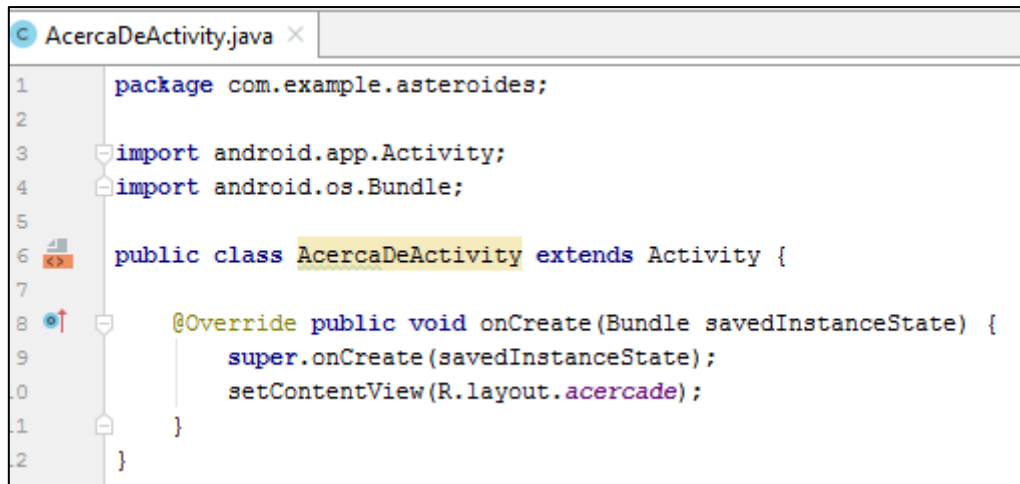


```

1  <?xml version="1.0" encoding="utf-8"?>
2  <resources>
3      <dimen name="margen_botones">30dp</dimen>
4      <dimen name="text_margin">20dp</dimen>
5  </resources>

```

Paso 3. Creamos ahora una nueva actividad, que será la responsable de visualizar esta vista. Para ello crea el fichero `AcercaDeActivity.java`, pulsando con el botón derecho sobre el nombre del paquete de la aplicación y seleccionando `New > Java Class`. En el campo `Name` introduce `AcercaDeActivity` y pulsa `Finish`. Reemplaza el código por:

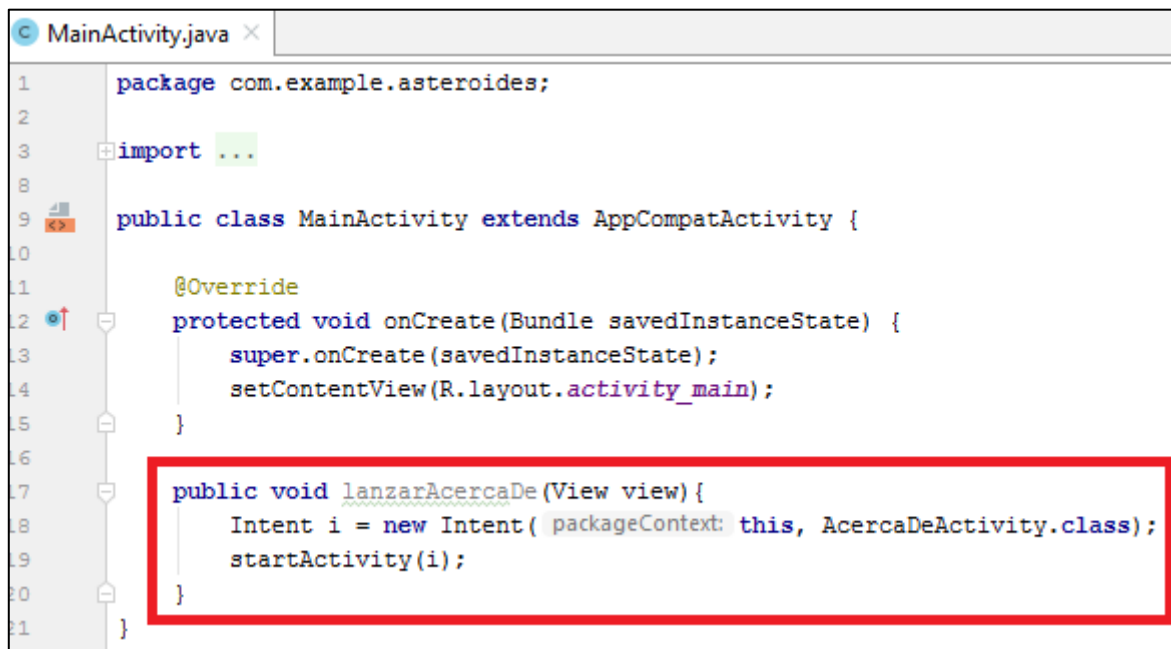


```

1  package com.example.asteroides;
2
3  import android.app.Activity;
4  import android.os.Bundle;
5
6  public class AcercaDeActivity extends Activity {
7
8      @Override public void onCreate(Bundle savedInstanceState) {
9          super.onCreate(savedInstanceState);
10         setContentView(R.layout.acercade);
11     }
12 }

```

Paso 4. Pasemos ahora a crear un método en la clase `Asteroides.java` que será ejecutado cuando sea pulsado el botón `Acerca de`.



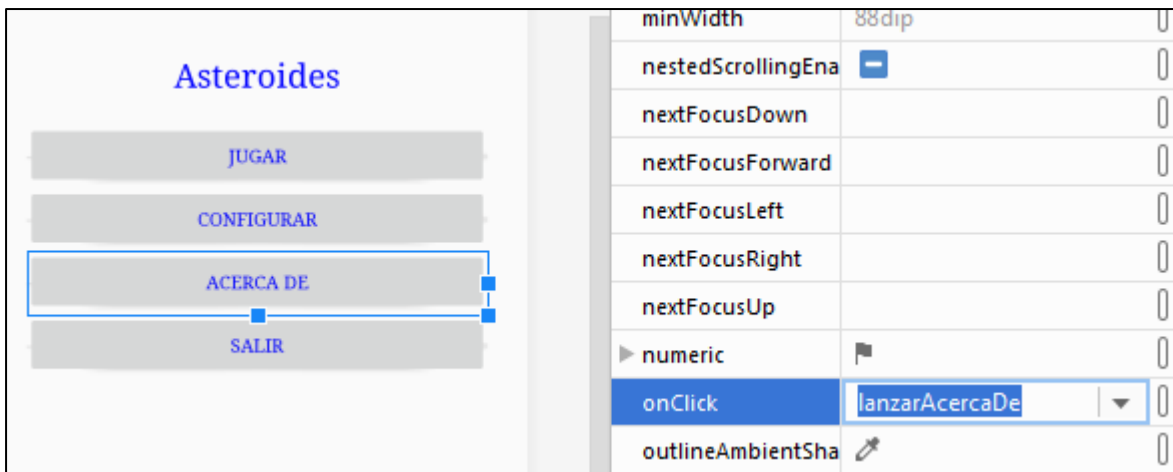
```

1  package com.example.asteroides;
2
3  import ...
4
5  public class MainActivity extends AppCompatActivity {
6
7      @Override
8      protected void onCreate(Bundle savedInstanceState) {
9          super.onCreate(savedInstanceState);
10         setContentView(R.layout.activity_main);
11     }
12
13     public void lanzarAcercaDe(View view) {
14         Intent i = new Intent( packageContext: this, AcercaDeActivity.class);
15         startActivity(i);
16     }
17 }

```

Nota sobre Java: Pulsa **Alt-Intro**, para que automáticamente se añadan los paquetes que faltan.

Paso 5. Para asociar este método al botón edita el `Layout activity_main.xml`. Selecciona la lengüeta `Design` y pulsa sobre el botón `Acerca de...` y en la vista `Properties` busca el atributo `onClick` e introduce el valor `lanzarAcercaDe`.



Se ha de hacer esto para las el **activity_main.xml** y para el **activity_main.xml (land)**.

Paso 6. Pasa a la edición xml pulsando en la lengüeta *main.xml* y observa como en la etiqueta `<Button>` correspondiente, se ha añadido el atributo:

`android:onClick="lanzarAcercaDe"`

```
<Button
    android:id="@+id/Button03"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:onClick="lanzarAcercaDe"
    android:text="Acerca De" />
```

NOTA: En caso de que exista algún recurso alternativo para el layout repite el mismo proceso.

Paso 7. Ejecuta ahora la aplicación y pulsa en el botón *Acerca de*. Observarás que el resultado no es satisfactorio ¿Qué ha ocurrido?

```
2019-10-29 19:19:22.549 15285-15285/? E/AndroidRuntime: FATAL EXCEPTION: main
Process: com.example.asteroides, PID: 15285
java.lang.IllegalStateException: Could not execute method for android:onClick
    at androidx.appcompat.app.AppCompatActivity$DeclaredOnClickListener.onClick(AppCompatActivity.java:402)
    at android.view.View.performClick(View.java:7339)
    at android.widget.TextView.performClick(TextView.java:14222)
    at android.view.View.performClickInternal(View.java:7305)
    at android.view.View.access$3200(View.java:245)
    at android.view.View$PerformClick.run(View.java:27787)
    at android.os.Handler.handleCallback(Handler.java:873)
    at android.os.Handler.dispatchMessage(Handler.java:99)
    at android.os.Looper.loop(Looper.java:214)
    at android.app.ActivityThread.main(ActivityThread.java:7078) <1 internal call>
    at com.android.internal.os.RuntimeInit$MethodAndArgsCaller.run(RuntimeInit.java:493)
    at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:975)
Caused by: java.lang.reflect.InvocationTargetException <1 internal call>
    at androidx.appcompat.app.AppCompatActivity$DeclaredOnClickListener.onClick(AppCompatActivity.java:397) <9 more...> <1 internal call> <2 more...>
Caused by: android.content.ActivityNotFoundException: Unable to find explicit activity class {com.example.asteroides/com.example.asteroides.AcercaDeActivity}; have you declared this activity in your AndroidManifest.xml?
    at android.app.Instrumentation.checkStartActivityResult(Instrumentation.java:12020)
    at android.app.Activity.startActivityForResult(Activity.java:4689)
    at androidx.fragment.app.FragmentActivity.startActivityForResult(FragmentActivity.java:676)
    at android.app.Activity.startActivityForResult(Activity.java:4647)
    at androidx.fragment.app.FragmentActivity.startActivityForResult(FragmentActivity.java:663)
    at android.app.Activity.startActivity(Activity.java:5008)
    at android.app.Activity.startActivity(Activity.java:4976)
    at com.example.asteroides.MainActivity.lanzarAcercaDe(MainActivity.java:19) <1 internal call> <10 more...> <1 internal call> <2 more...>
```

El problema es que toda actividad que ha de ser lanzada por una aplicación ha de ser registrada en el fichero **AndroidManifest.xml**. Para registrar la actividad, abre **AndroidManifest.xml**. Añade el siguiente texto dentro de la etiqueta `<application ...>` `</application>`:

```
<activity android:name=".AcercaDeActivity" android:label="Acerca de ..." />
```

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.asteroides">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Asteroides"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

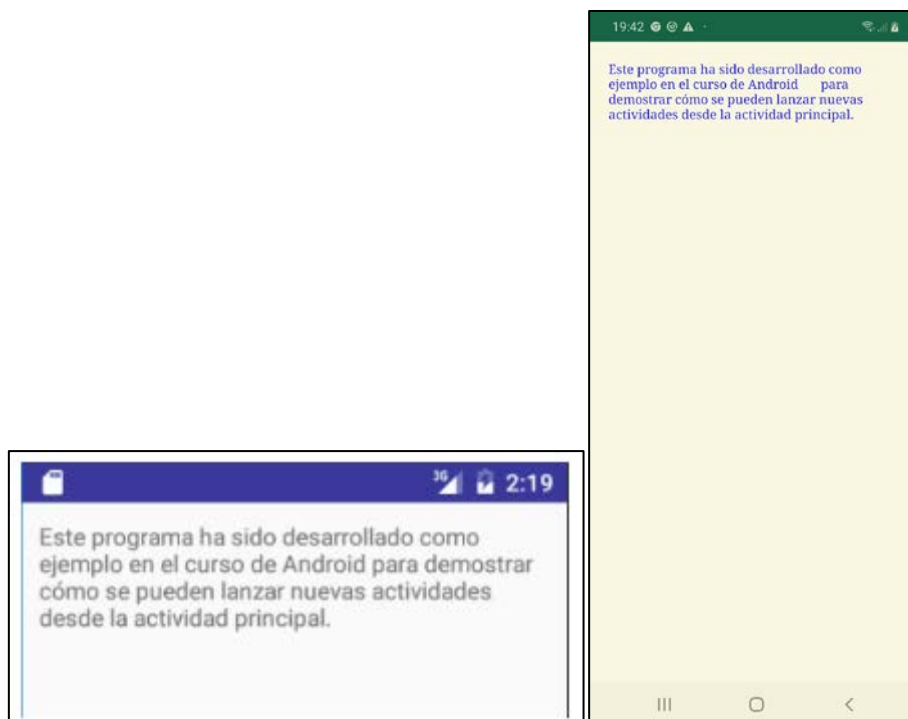
        <activity android:label="Acerca de ..." android:name=".AcercaDeActivity" />

    </application>

</manifest>

```

Paso 8. Ejecuta de nuevo el programa. El resultado ha de ser similar al mostrado a continuación:



La vista mostrada en el ejemplo anterior no parece muy atractiva. Tratemos de mejorarla aplicando un tema. Como vimos en el capítulo anterior un tema es una colección de estilos que define el aspecto de una actividad o aplicación. Puedes utilizar alguno de los temas disponibles en Android o crear el tuyo propio.

Paso 9. En este caso utilizaremos uno de los de Android. Para ello abre *AndroidManifest.xml* e introduce la línea subrayada:

```

<activity android:label="Acerca de ..." android:name=".AcercaDeActivity"
    android:theme="@style/Theme.AppCompat.Light.Dialog" />

```

Paso 10. Ejecuta de nuevo el programa y observa cómo mejora la apariencia:



Parte II: Un escuchador de evento por código

Como acabamos de ver en un layout podemos definir el atributo XML `android:onClick` que nos permite indicar un método que será ejecutado al hacer click en una vista. A este método se le conoce como escuchador de evento. Resulta muy sencillo y además está disponible en cualquier descendiente de la clase `View`. Sin embargo esta técnica presenta dos inconvenientes. Solo está disponible para el evento `onClick()`. La clase `View` tiene otros eventos (`onLongClick()`, `onFocusChange()`, `onKey()`,...) para los que no se han definido un atributo xml. Entonces, ¿qué hacemos si queremos definir un evento distinto de `onClick()`. La respuesta la encontrarás en este ejercicio:

Paso 1. Abre la clase `Asteroides.java` y añade las líneas que aparecen subrayadas:

```
public class MainActivity extends AppCompatActivity {

    private Button bAcercaDe;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        bAcercaDe = findViewById(R.id.button03);
        bAcercaDe.setOnClickListener(new View.OnClickListener() {
            public void onClick(View view) {
                lanzarAcercaDe( view: null);
            }
        });

        public void lanzarAcercaDe(View view){
            Intent i = new Intent( packageContext: this, AcercaDeActivity.class);
            startActivity(i);
        }
    }
}
```

Nota sobre Java: Pulsa **Alt-Intro** para que automáticamente se añadan los paquetes que faltan. Para la clase OnClickListener selecciona android.view.View.OnClickListener.

Paso 2. Elimina el atributo añadido al botón:

android:onClick="lanzarAcercaDe"

```
<Button
    android:id="@+id/Button03"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Acerca De"/>

```

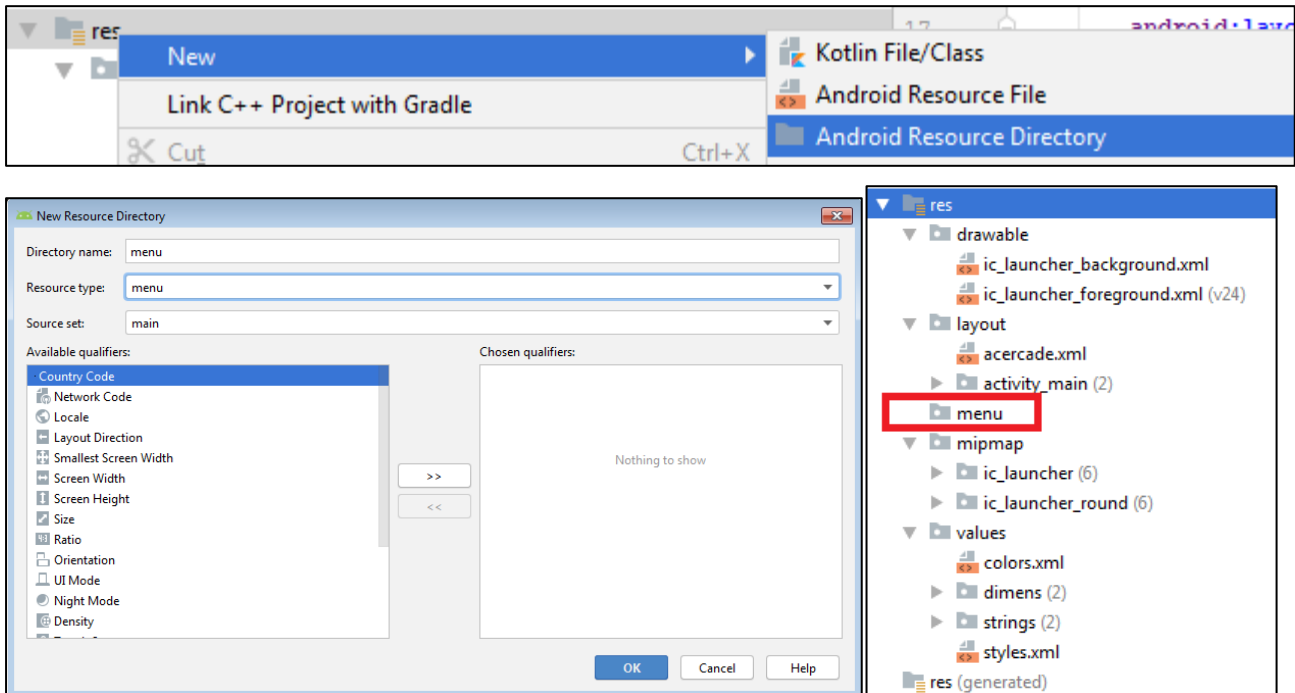
Paso 3. Verifica que el resultado es el mismo en ambos casos.

NOTA: No es conveniente que en tus actividades incluyas un botón para cerrarlas. Un dispositivo Android siempre dispone de la tecla «retorno», que tiene la misma función.

Parte IV: Añadiendo un menú a una actividad

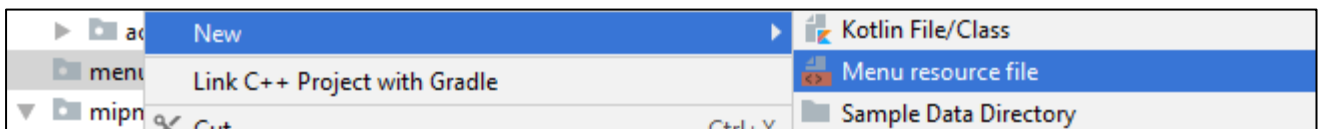
Podemos asignar un menú a nuestra actividad de forma muy sencilla.

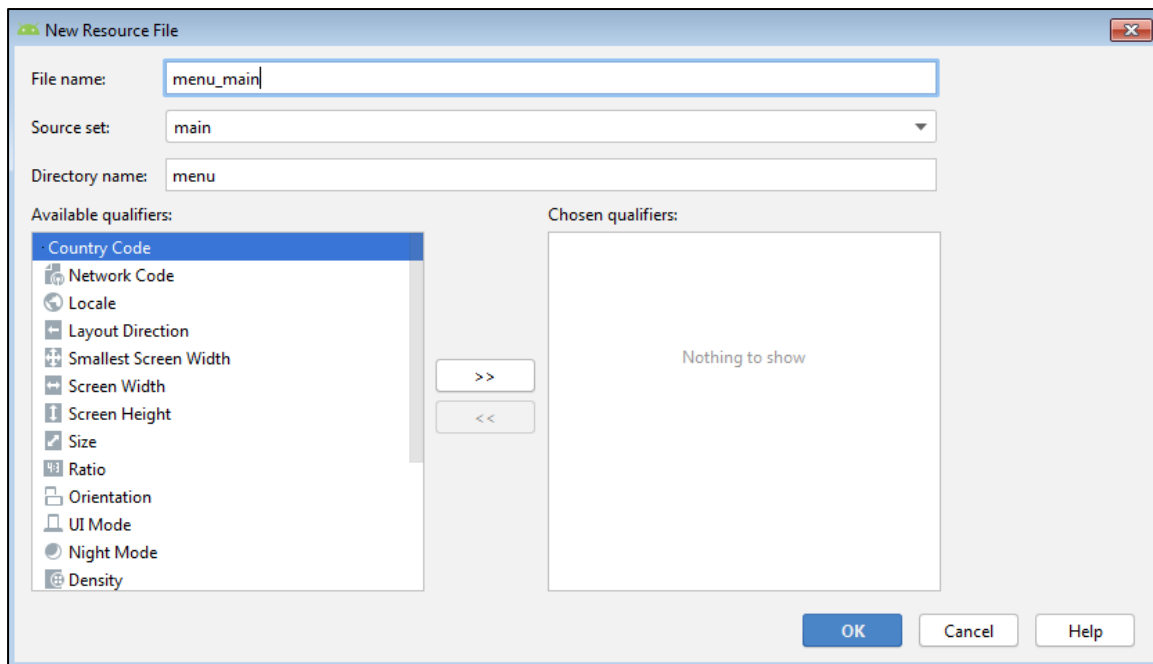
Paso 1. Abre el proyecto Asteroides. Crea una carpeta de recursos Android llamada menú



Paso 2. Crea un nuevo menú, usa File > New > Android resource file. En el campo File name: selecciona menu_main y en el campo Resource type: selecciona Menu.

Si desarrollas Mis Lugares abre el fichero res / menu / menu_main.xml.





NOTA: El fichero de menú se crea automáticamente si seleccionas una actividad de tipo: Basic Activity o Scrolling Activity.

Paso 3. Reemplaza el contenido que se muestran a continuación:

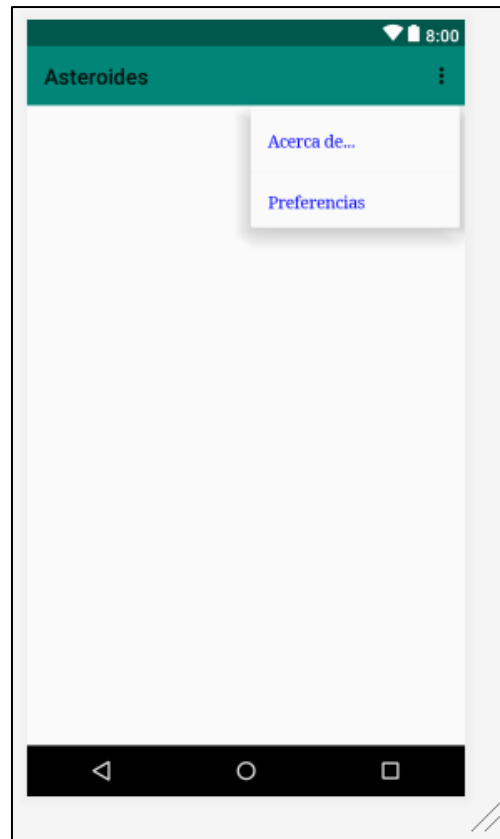
```

1  <?xml version="1.0" encoding="utf-8"?>
2  <menu xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:app="http://schemas.android.com/apk/res-auto">
4      <item android:id="@+id/action_settings"
5          android:title="Preferencias"
6          android:icon="@android:drawable/ic_menu_preferences"
7          android:orderInCategory="100"
8          app:showAsAction="never"/>
9      <item android:title="Acerca de..."
10         android:id="@+id/acercaDe"
11         android:icon="@android:drawable/ic_menu_info_details"/>
12  </menu>

```

Como puedes ver cada ítem de menú tiene tres atributos principales: id que permite identificarlo desde el código, title, para asociarle un texto e icon, para asociarle un icono. Los otros dos atributos se aplican cuando el menú actúa como una barra de acciones y serán explicados en el próximo punto.

Paso 4. A continuación se muestra la apariencia de este menú. Esta apariencia puede cambiar dependiendo de la versión de Android.

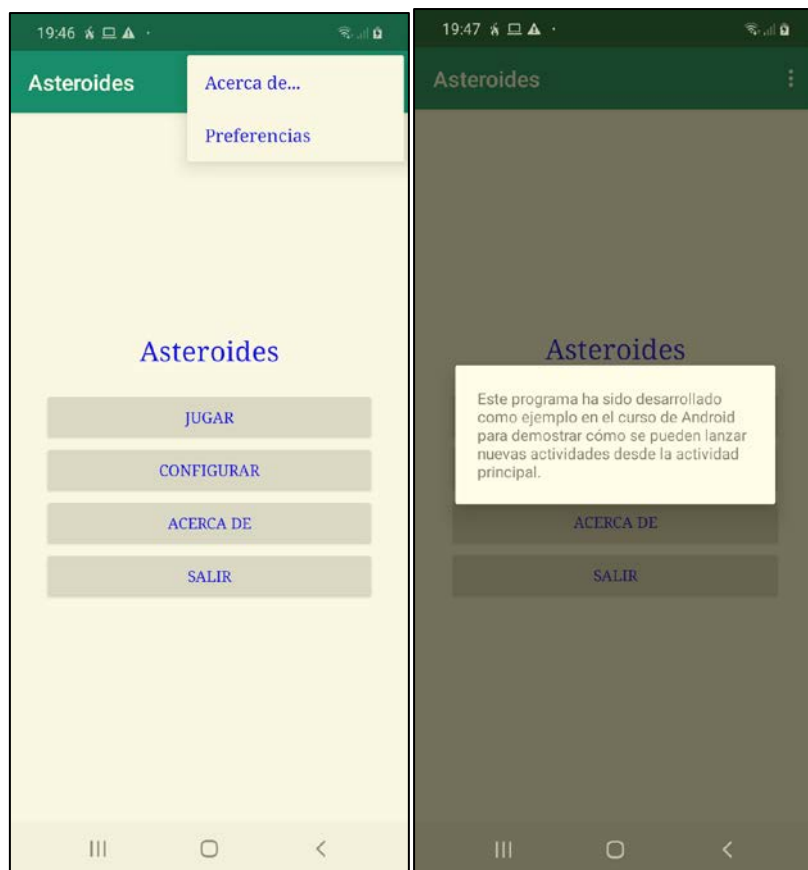


Paso 5. Para activar el menú, has de introducir el siguiente código en la actividad que muestra el menú. Posiblemente solo tengas que incluir el texto subrayado.

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.menu_main, menu);
    return true; /** true -> el menú ya está visible */
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    int id = item.getItemId();
    if (id == R.id.action_settings) {
        return true;
    }
    if (id == R.id.acercaDe) {
        lanzarAcercaDe( view: null);
        return true;
    }
    return super.onOptionsItemSelected(item);
}
```

Paso 6. Ejecuta el programa y pulsa en el icono del menú en la esquina superior derecha. Han de aparecer los dos ítems de menú. Selecciona Acerca de... para pasar a la actividad correspondiente.



Parte V: Añadiendo una barra de acciones a nuestra aplicación.

Paso 1. Reemplaza el contenido del fichero `res/menu/menu_main.xml` por:

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <menu xmlns:android="http://schemas.android.com/apk/res/android"
3        xmlns:app="http://schemas.android.com/apk/res-auto"
4        xmlns:tools="http://schemas.android.com/tools"
5        tools:context=".MainActivity">
6      <item android:id="@+id/action_settings"
7            android:title="Configuración"
8            android:icon="@android:drawable/ic_menu_preferences"
9            android:orderInCategory="5"
10           app:showAsAction="never"/>
11      <item android:title="Acerca de..."
12            android:id="@+id/acercaDe"
13            android:icon="@android:drawable/ic_menu_info_details"
14            android:orderInCategory="10"
15            app:showAsAction="ifRoom|withText"/>
16      <item android:title="Buscar"
17            android:id="@+id/menu_buscar"
18            android:icon="@android:drawable/ic_menu_search"
19            android:orderInCategory="115"
20            app:showAsAction="always|collapseActionView"/>
21    </menu>

```

En este fichero se define los iconos y las acciones a mostrar en la barra. Las acciones que indiquen en el atributo `showAsAction` la palabra `always` se mostrarán siempre, sin importar si caben o no. El uso de estas acciones debería limitarse, lo ideal es que haya una o dos, ya que al forzar que se visualicen todas podrían verse incorrectamente. Las acciones que indiquen `ifRoom` se mostrarán en la barra de acciones si hay espacio disponible, y se moverán al menú de *Overflow* si no lo hay. En esta categoría se deberían encontrar la mayoría de las acciones. Si se indica `never`, la acción nunca se mostrará en la barra de acciones, sin importar el espacio disponible. En este grupo se deberían situar acciones

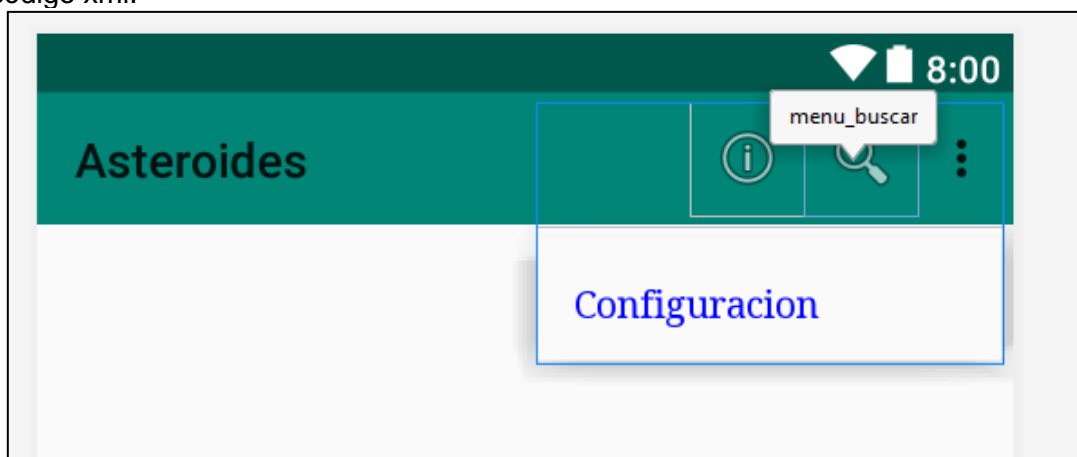
como modificar las preferencias, que deben estar disponibles para el usuario, pero no visibles en todo momento.

Las acciones se ordenan de izquierda a derecha según lo indicado en `orderInCategory`, con las acciones con un número más pequeño más a la izquierda. Si no caben todas las acciones en la barra, las que tienen un número mayor se mueven al menú de *Overflow*.

Paso 2. Ejecuta la aplicación. Podrás ver como aparece la barra de acciones en la parte de arriba, con los botones que hemos definido.



Android Studio incorpora un editor visual de menús que nos permite crear menús sin necesidad de escribir código xml.



Parte VI: Creación de iconos personalizados para Asteroides

Veamos un ejemplo práctico de cómo crear un icono: El diseñador gráfico de nuestra empresa nos acaba de pasar el icono asociado para iniciar la aplicación que estamos diseñando (launcher icon).

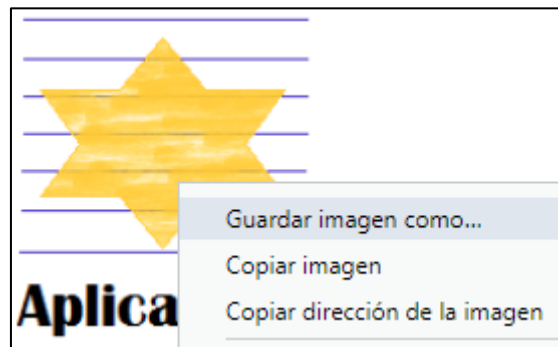


Aplicación

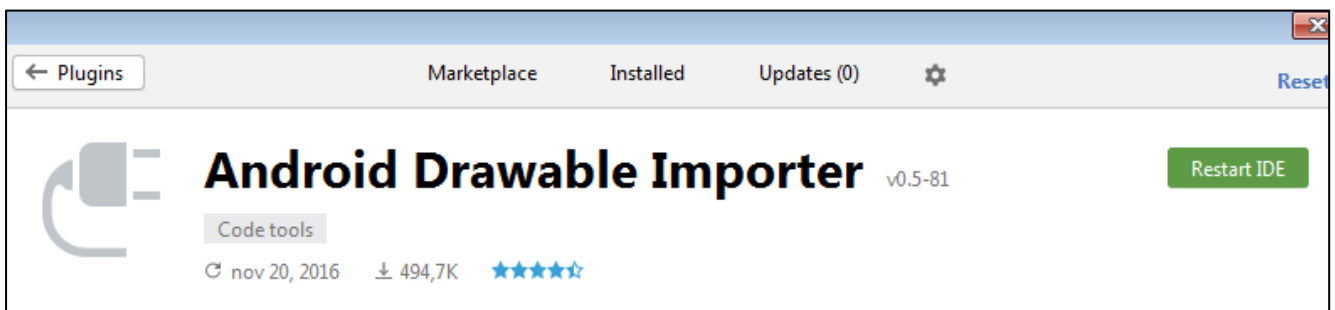
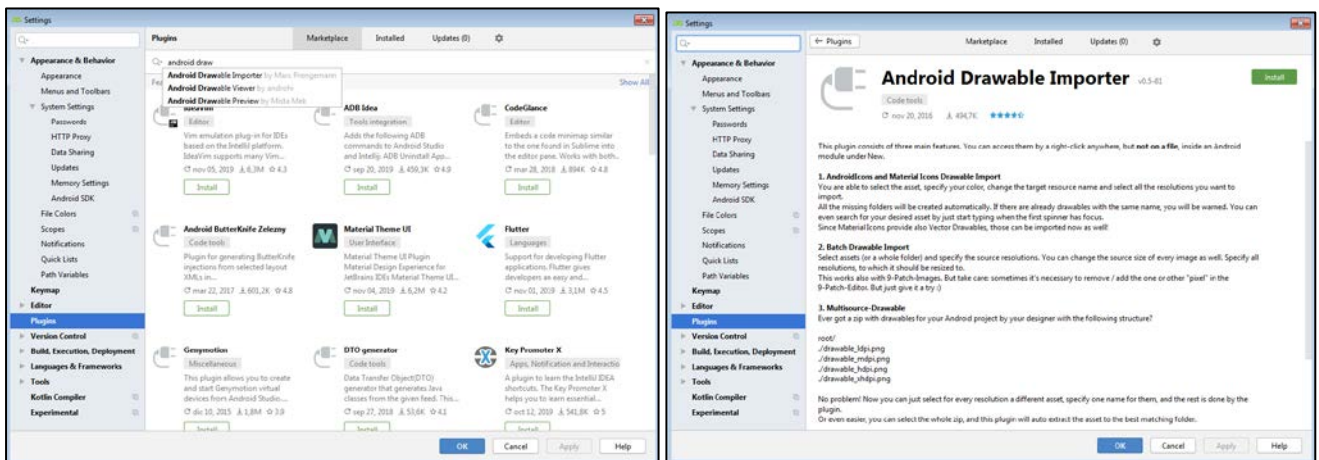
(Encontrarás este icono en el moodle)

Para asignarlo a una aplicación realiza los siguientes pasos:

Paso 1. Descarga el gráfico anterior del Moodle debajo de este enunciado y llámale *icon.png*.

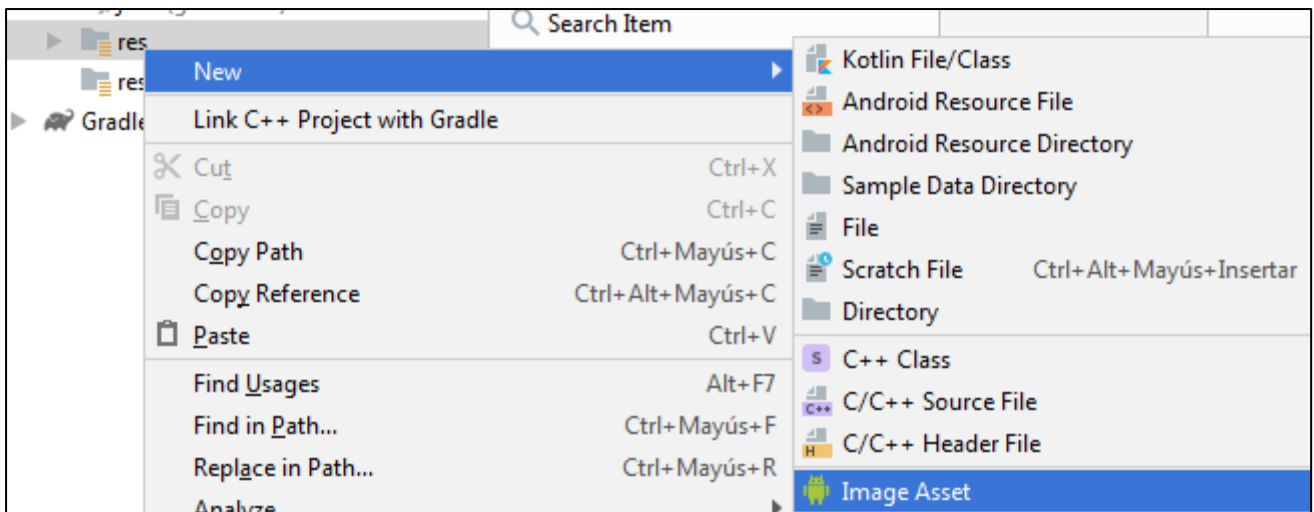


Paso 2. Instala el plugin Android Drawable Importer. Ve a File/Settings/Plugins y busca el plugin. Instálalo y reinicia Android Studio.

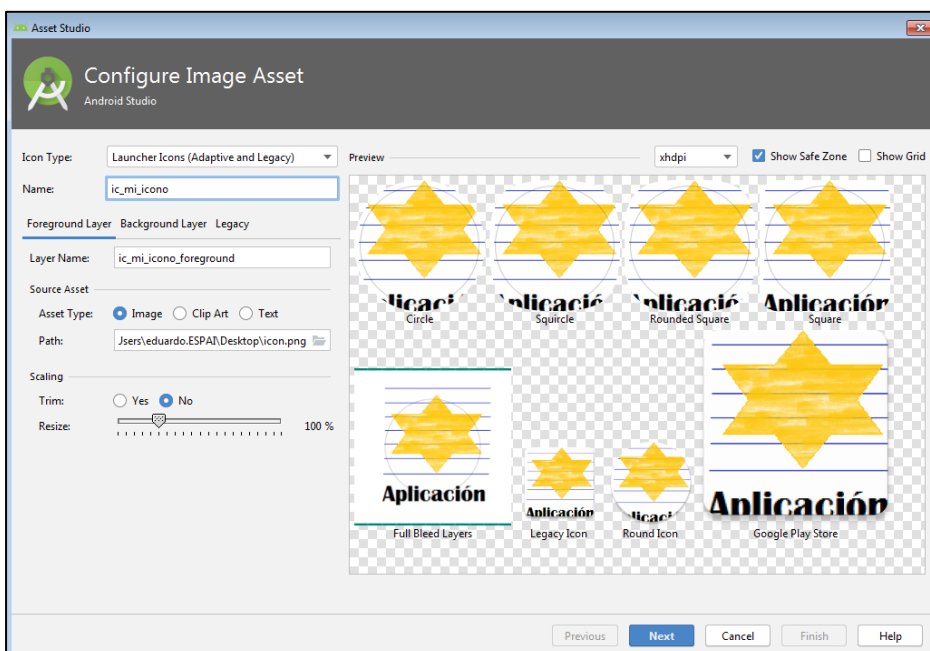
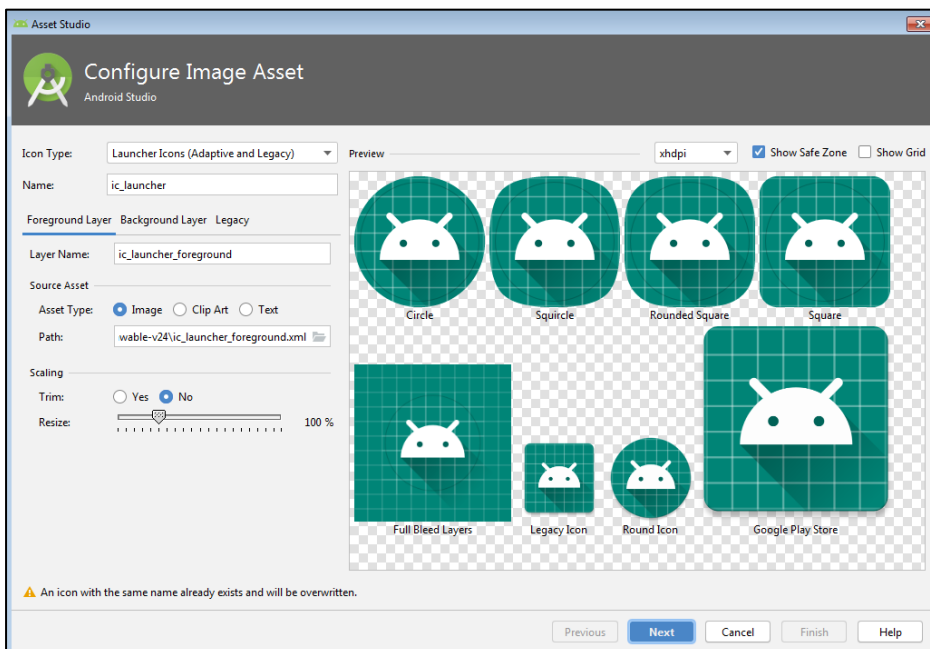


Si no te aparece en pluguins es probable que tu versión de Android ya incluya el pluguin.

Paso 3. En el explorador proyecto pulsa con el botón derecho sobre la carpeta res y selecciona File / New / Image Assets.

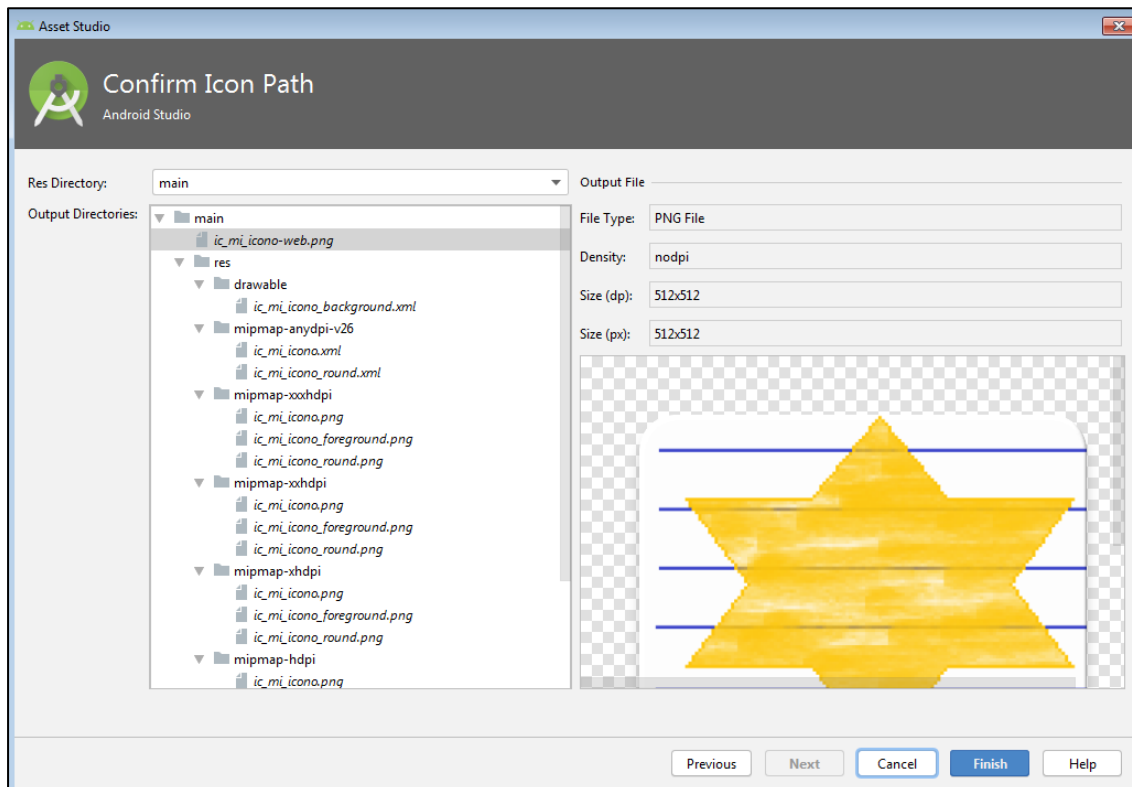


Paso 4. En el campo Name: introduce ic_mi_icono; en Asset Type: Image y en Path: indica el fichero descargado en el paso 1. El resultado ha de ser similar a:

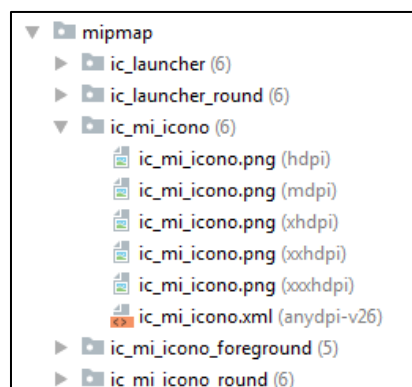


En la parte inferior de la página podrás previsualizar cómo quedarán las imágenes para diferentes densidades gráficas.

Paso 5. Pulsa el botón Next y en la siguiente pantalla pulsa Finish.



Paso 6. Verifica como dentro de la carpeta res/mipmap se han creado 5 recursos alternativos para diferentes densidades gráficas:



Paso 7. El resultado es aceptable para algunas opciones. Pero en algunos casos el texto no se lee o unas líneas horizontales quedan más gruesas que otras. Puede ser interesante retocar estos ficheros png usando un editor de gráficos. Por ejemplo, retoca los iconos hdpi y mdpi para que se pueda leer más claramente el texto “Aplicación” y redibuja las líneas horizontales para que tenga el mismo grosor y separación.

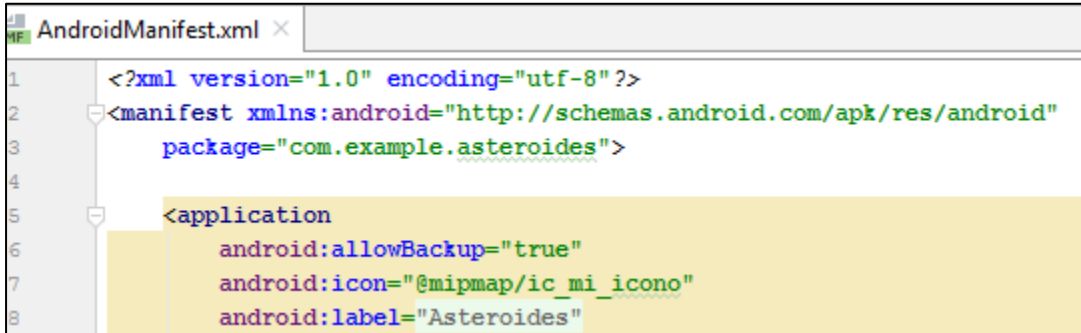


Paso 8. Para que el nuevo icono sea utilizado como lanzador de nuestra aplicación, abre AndroidManifest.xml y reemplaza el valor del atributo icon como se muestra a continuación:

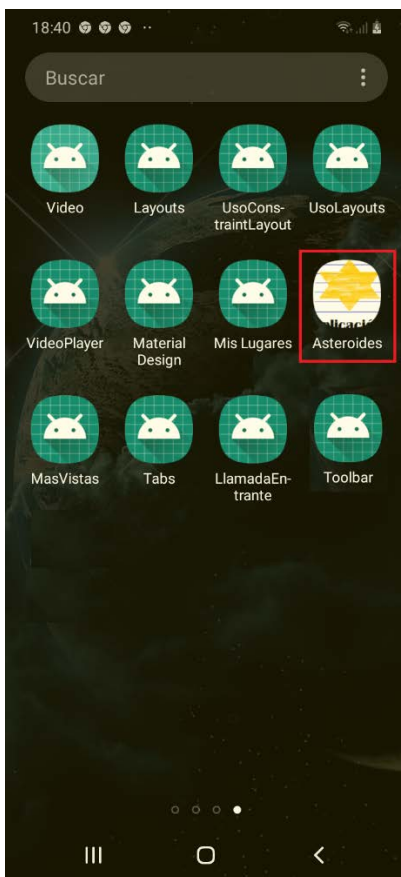
```
<application
```

```
    android:allowBackup="true"
```

```
    android:icon="@mipmap/ic_launchermi_icono"
```



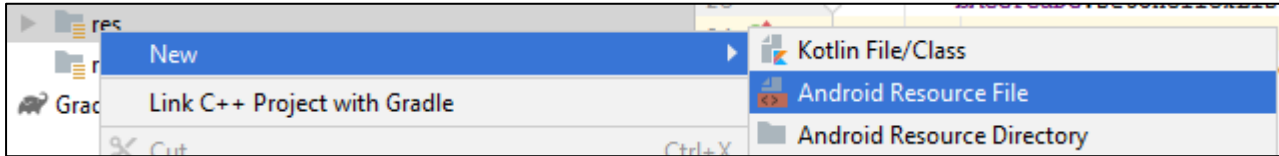
Paso 9. Visualiza el resultado instalando la aplicación en diferentes dispositivos con diferentes densidades gráficas.



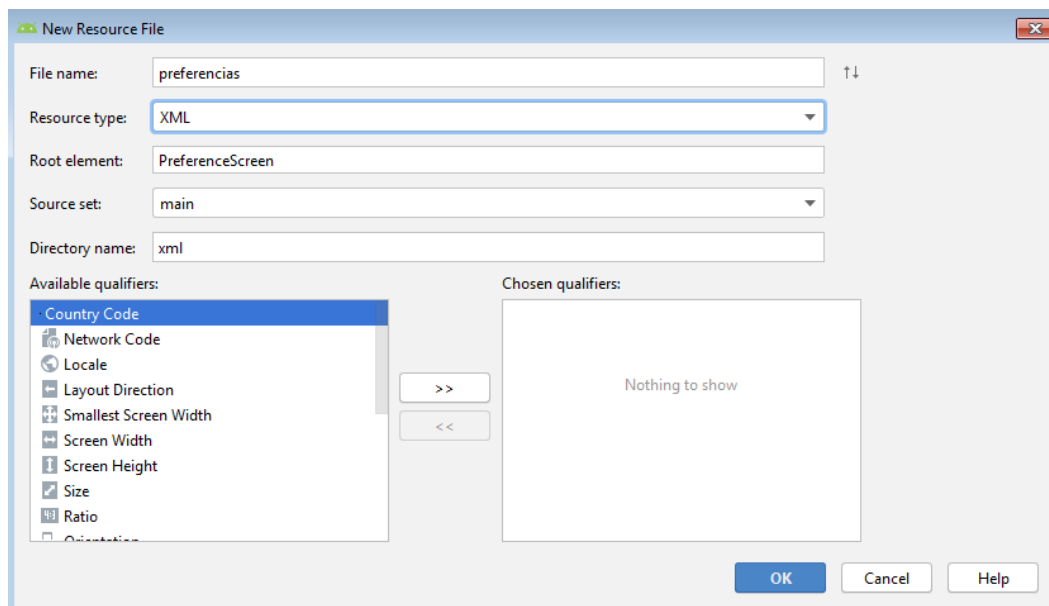
Parte VII: Añadiendo preferencias Asteroides

Paso 1. Abre el proyecto Asteroides.

Paso 2. Pulsa con el botón derecho sobre la carpeta *res* y selecciona la opción *New > Android resource File*.



Paso 3. Completa los campos File name: preferencias y Resource type: XML. Se creará el fichero *res/xml/preferencias.xml*.



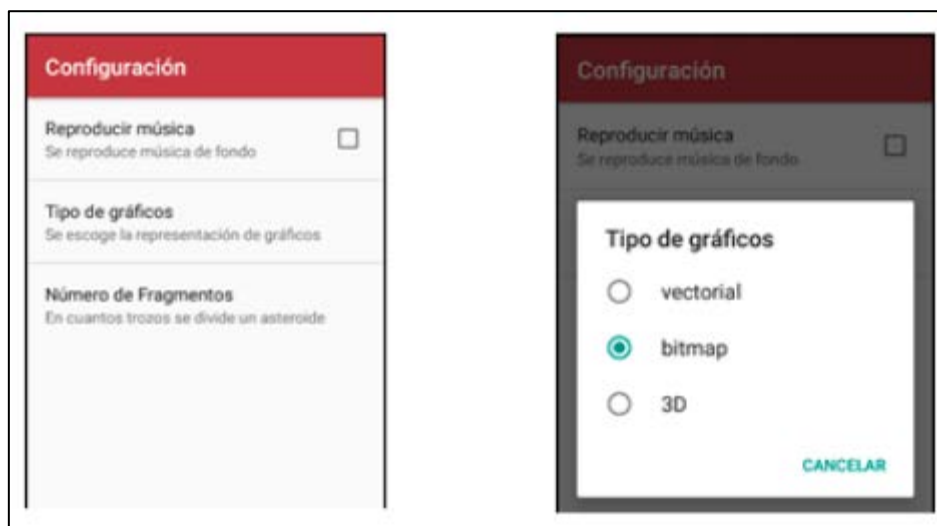
Paso 4. Edita el fichero e introduce el siguiente código:


```

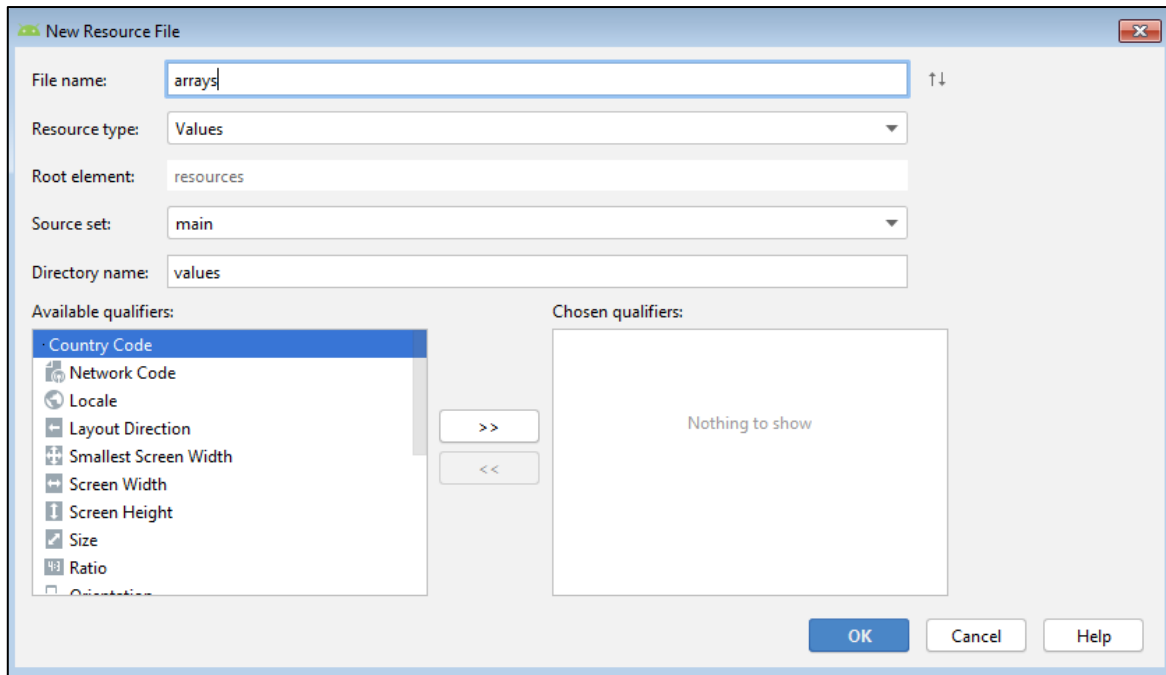
1  <?xml version="1.0" encoding="utf-8"?>
2  <PreferenceScreen
3      xmlns:android="http://schemas.android.com/apk/res/android"
4      android:key="preferencias_principal">
5      <CheckBoxPreference
6          android:key="musica"
7          android:title="Reproducir música"
8          android:summary="Se reproduce música de fondo"/>
9      <ListPreference
10         android:key="graficos"
11         android:title="Tipo de gráficos"
12         android:summary="Se escoge la representación de gráficos"
13         android:entries="@array/tiposGraficos"
14         android:entryValues="@array/tiposGraficosValores"
15         android:defaultValue="1"/>
16      <EditTextPreference
17         android:key="fragmentos"
18         android:title="Número de Fragmentos"
19         android:summary="En cuantos trozos se divide un asteroide"
20         android:inputType="number"
21         android:defaultValue="3"/>
22  </PreferenceScreen>

```

El significado de cada etiqueta y atributo se descubre fácilmente si observas el resultado obtenido que se muestra a continuación. El atributo `inputType` permite configurar el tipo de teclado que se mostrará para introducir el valor. Coinciden con el atributo de `EditText`. Para ver los posibles valores consultar developer.android.com/reference/android/widget/TextView.html#attr_android:inputType



Paso 5. Para almacenar los valores del desplegable, has de crear el fichero `/res/values/arrays.xml` con el siguiente contenido. Para ello pulsa con el botón derecho sobre la carpeta `res` y selecciona la opción `New > Android resource file`. Completa los campos `File name: arrays` y `Resource type: values`.



```

arrays.xml
1  <?xml version="1.0" encoding="utf-8"?>
2  <resources>
3    <string-array name="tiposGraficos">
4      <item>vectorial</item>
5      <item>bitmap</item>
6      <item>3D</item>
7    </string-array>
8    <string-array name="tiposGraficosValores">
9      <item>0</item>
10     <item>1</item>
11     <item>2</item>
12   </string-array>
13 </resources>

```

Paso 6. Crea ahora una nueva clase *Preferencias.java* con el siguiente código:

```

Preferencias.java
package com.example.asteroides;

import android.app.Activity;
import android.os.Bundle;
import android.preference.PreferenceFragment;

public class Preferencias extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        getFragmentManager().beginTransaction().
            replace(android.R.id.content, new MyPreferenceFragment()).commit();
    }

    public static class MyPreferenceFragment extends PreferenceFragment {
        @Override
        public void onCreate(final Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            addPreferencesFromResource(R.xml.preferencias);
        }
    }
}

```

Nota sobre Java: Pulsa Alt-Intro para que automáticamente se añadan los paquetes que faltan.

NOTA: Desde el nivel de API 11, el método `addPreferencesFromResource()` se ha marcado como obsoleto. En lugar de usar la clase `PreferenceActivity`, se recomienda utilizar `PreferenceFragment`. Sin embargo, si estamos trabajando con un nivel mínimo de API inferior al 11 (en nuestro caso el 9), no disponemos de esta clase. Cuando un método o clase está marcado como obsoleto, se nos indica que hay otra forma más moderna de hacerlo, incluso que en un futuro es posible que deje de estar incluido en el API. Pero esto es algo que nunca ha pasado. En Android es frecuente seguir utilizando métodos marcados como obsoletos. En el siguiente apartado se explica el uso de `PreferenceFragment`.

NOTA: Recientemente, Google ha creado las librerías compatibilidad `preference-v7` y `preference-v14` donde se define `PreferenceFragmentCompact` una versión de `PreferenceFragment` compatible con versiones anteriores. El uso de `PreferenceFragmentCompact` se muestra en el Gran Libro de Android Avanzado.

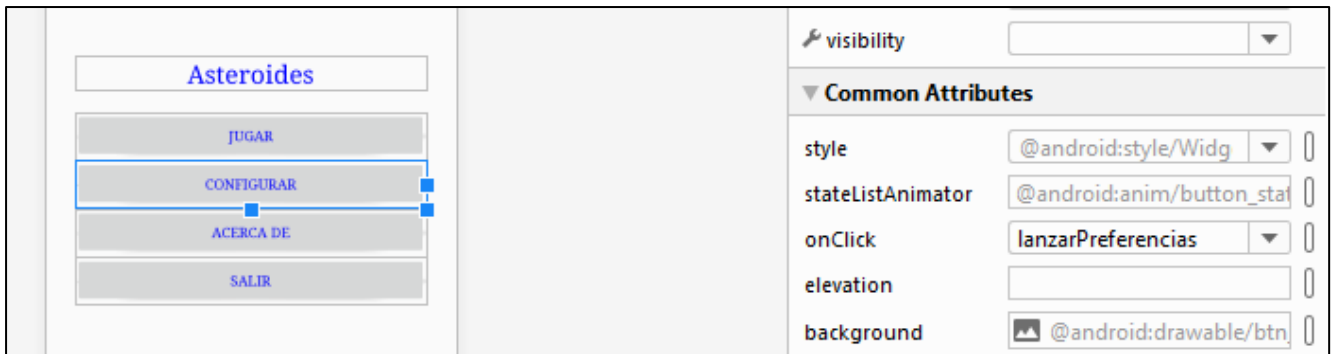
Paso 7. No hay que olvidar registrar toda nueva actividad en **AndroidManifest.xml**.



Paso 8. Añade a `MainActivity.java` el método `lanzarPreferencias()`. Este método ha de tener el mismo código que `lanzarAcercaDe()` pero lanzando la actividad `Preferencias`. En el Layout `activity_main.xml` añade al botón con texto “Configurar” en el atributo `onClick` el valor `lanzarPreferencias`.

```
public void lanzarAcercaDe(View view) {
    Intent i = new Intent( packageContext: this, AcercaDeActivity.class);
    startActivity(i);
}

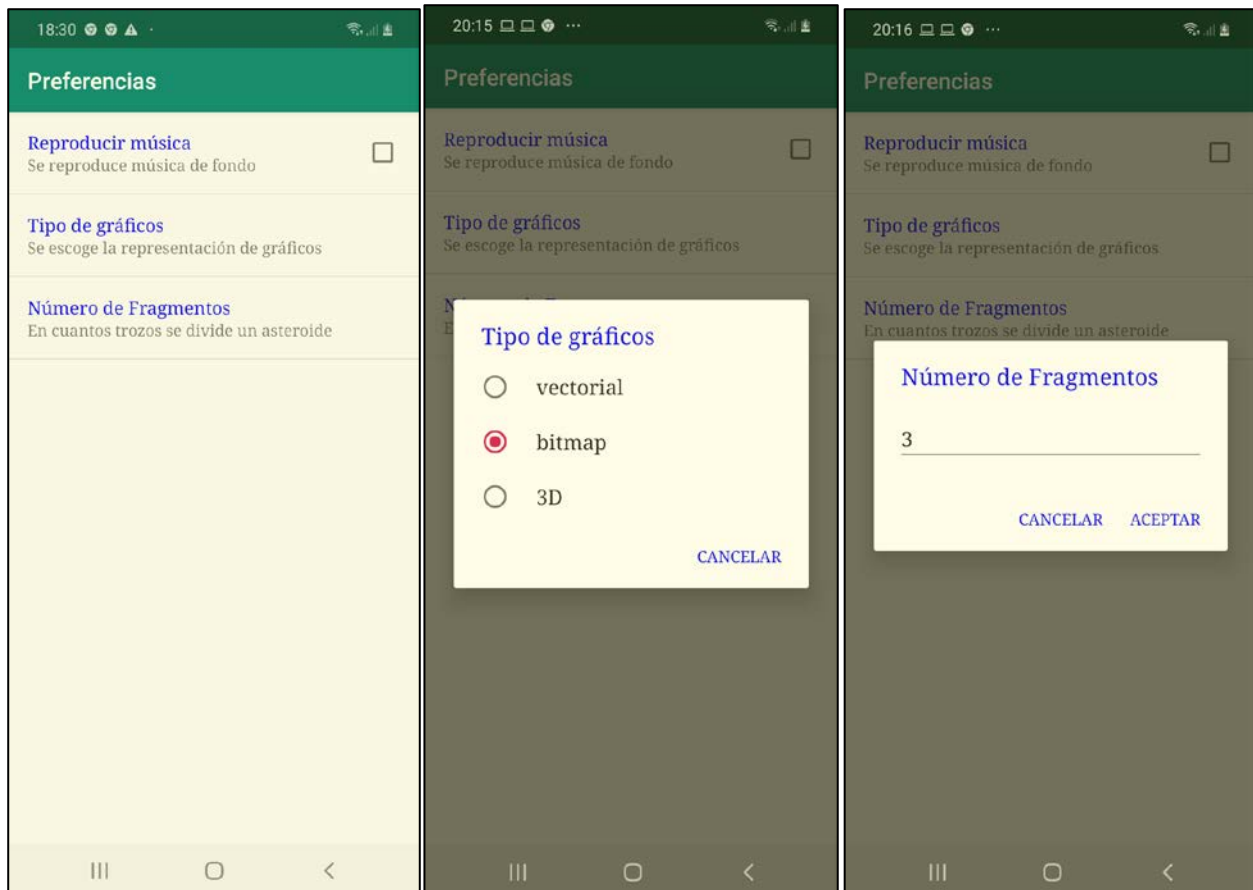
public void lanzarPreferencias(View view) {
    Intent i = new Intent( packageContext: this, Preferencias.class);
    startActivity(i);
}
```



Paso 9. Para activar la configuración desde la opción de menú añade el siguiente código en el fichero MainActivity.java en el método onOptionsItemSelected() dentro del switch:

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    int id = item.getItemId();
    if (id == R.id.acercaDe) {
        lanzarAcercaDe( view: null);
        return true;
    }
    if (id == R.id.action_settings) {
        lanzarPreferencias( view: null);
        return true;
    }
    return super.onOptionsItemSelected(item);
}
```

Paso 10. Arranca la aplicación y verifica que puedes lanzar las preferencias mediante las dos alternativas.



Parte VIII: Organizando preferencias

Paso 1. Crea una nueva lista de preferencias `<PreferenceScreen>` dentro de la lista de preferencias del fichero `res/xml/preferencias.xml`.



```
1  <?xml version="1.0" encoding="utf-8"?>
2  <PreferenceScreen
3      xmlns:android="http://schemas.android.com/apk/res/android"
4      android:key="preferencias_principal">
5      <CheckBoxPreference
6          android:key="musica"
7          android:title="Reproducir música"
8          android:summary="Se reproduce música de fondo"/>
9      <ListPreference
10         android:key="graficos"
11         android:title="Tipo de gráficos"
12         android:summary="Se escoge la representación de gráficos"
13         android:entries="@array/tiposGraficos"
14         android:entryValues="@array/tiposGraficosValores"
15         android:defaultValue="1"/>
16      <EditTextPreference
17         android:key="fragmentos"
18         android:title="Número de Fragmentos"
19         android:summary="En cuantos trozos se divide un asteroide"
20         android:inputType="number"
21         android:defaultValue="3"/>
22      <PreferenceScreen
23         android:title="Modo multijugador">
24      </PreferenceScreen>
25  </PreferenceScreen>
```

Paso 2. Asígnale al parámetro `android:title` el valor "Modo multijugador".

Paso 3. Crea tres elementos dentro de esta lista: *Activar multijugador*, *Máximo de jugadores* y *Tipo de conexión*. Para este último han de poder escogerse los valores: *Bluetooth*, *Wi-Fi* e *Internet*.

```

<PreferenceScreen
    android:title="Modo multijugador"
    android:key="Modo multijugador">
    <CheckBoxPreference
        android:key="activar_multijugador"
        android:title="Activar modo multijugador"
        android:summary="Si se activa el modo multijugador"/>
    <EditTextPreference
        android:key="maxjugadores"
        android:title="Maximo de Jugadores"
        android:summary="Maximo numero de jugadores"
        android:inputType="number"
        android:defaultValue="3"/>
    <ListPreference
        android:key="tipoconexion"
        android:title="Tipo de conexion"
        android:summary="Tipos de conexion Internet"
        android:entries="@array/tiposConexion"
        android:entryValues="@array/tiposConexionValores"
        android:defaultValue="1"/>
</PreferenceScreen>

```

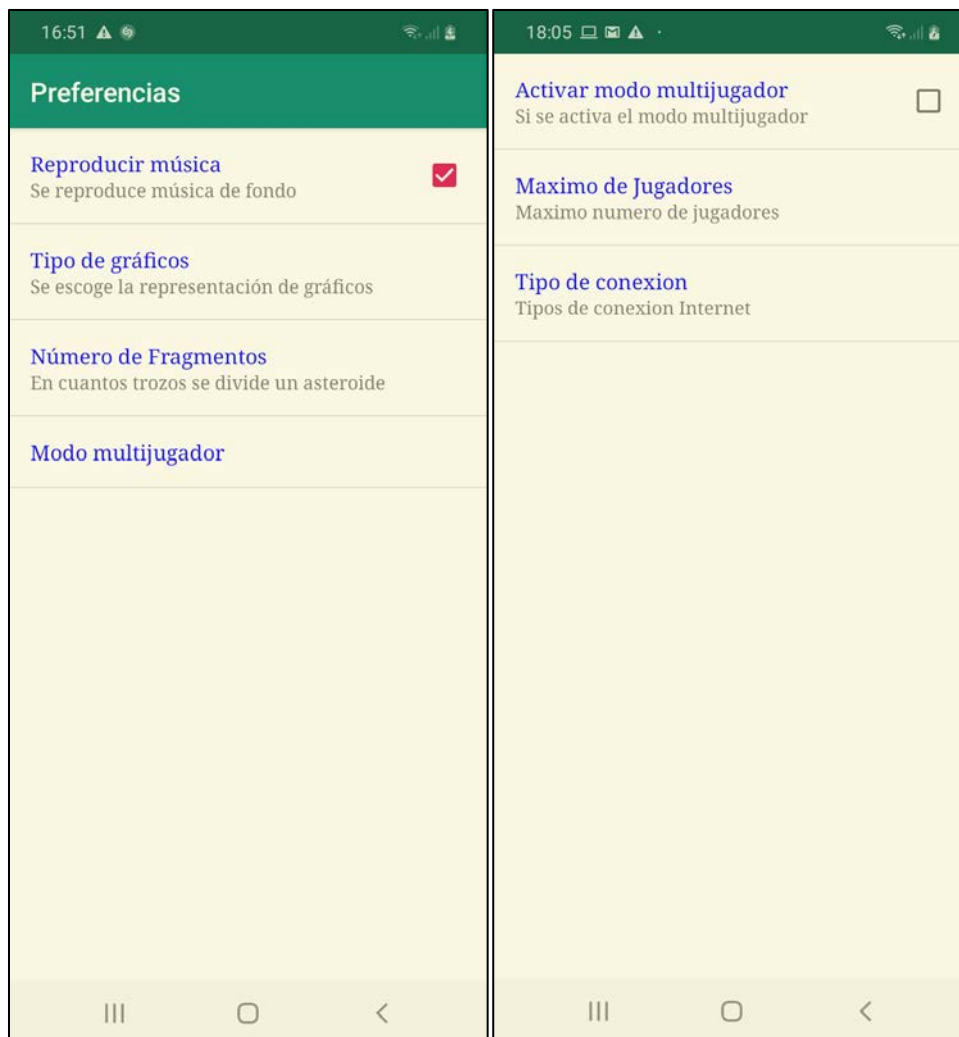
```

<string-array name="tiposConexion">
    <item>Bluetooth</item>
    <item>Wi-Fi</item>
    <item>Internet</item>
</string-array>

<string-array name="tiposConexionValores">
    <item>0</item>
    <item>1</item>
    <item>2</item>
</string-array>

```

Paso 4. Visualiza el resultado.



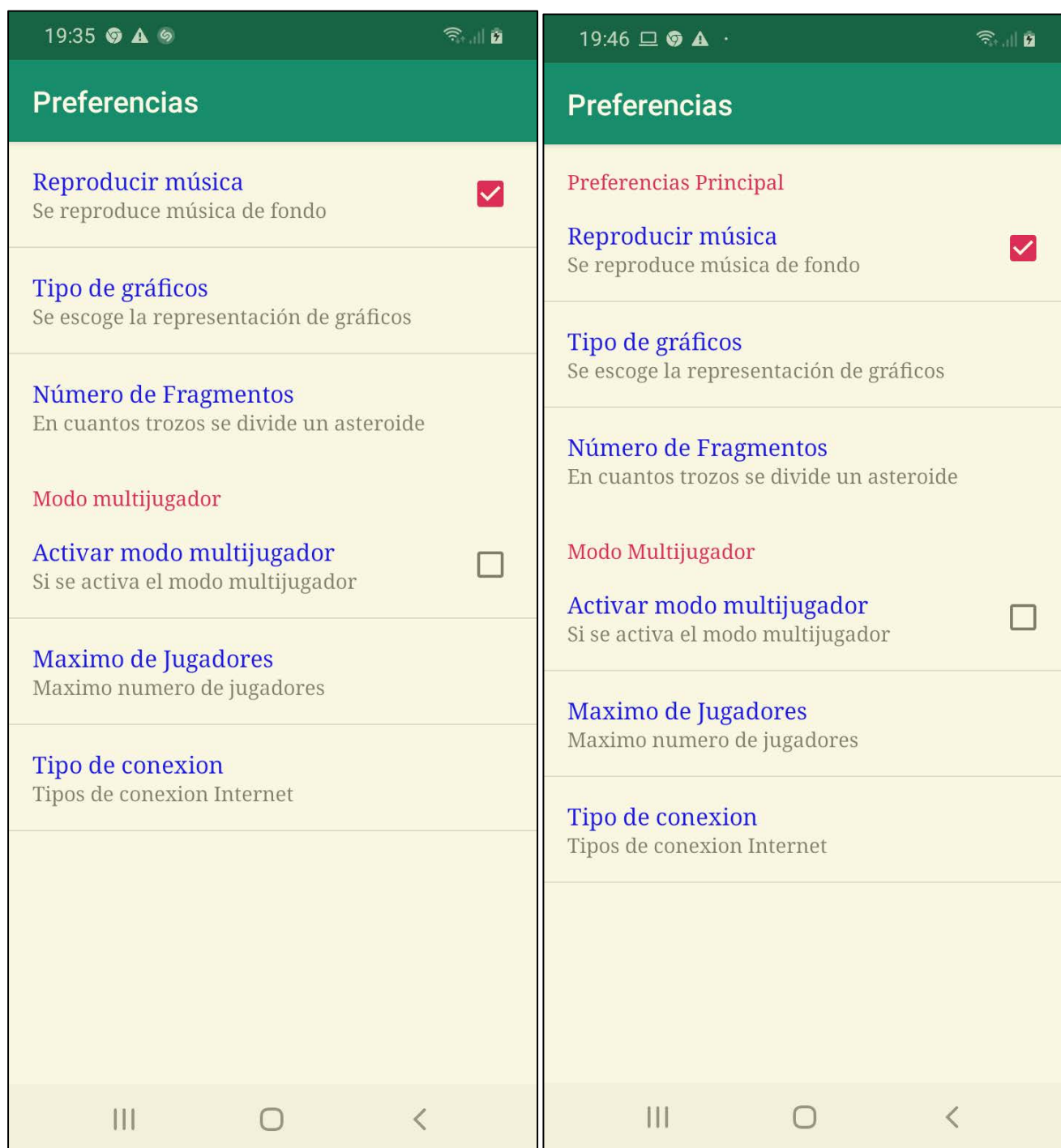
Paso 5. Otra opción para organizar las preferencias consiste en agruparlas por categorías. Con esta opción se visualizarán en la misma pantalla, pero separadas por grupos. Has de seguir el siguiente esquema:

```

<PreferenceScreen
    <CheckBoxPreference .../>
    <EditTextPreference .../>
    ...
    <PreferenceCategory android:title="Modo multijugador">
        <CheckBoxPreference .../>
        <EditTextPreference .../>
        ...
    </PreferenceCategory>
</PreferenceScreen>

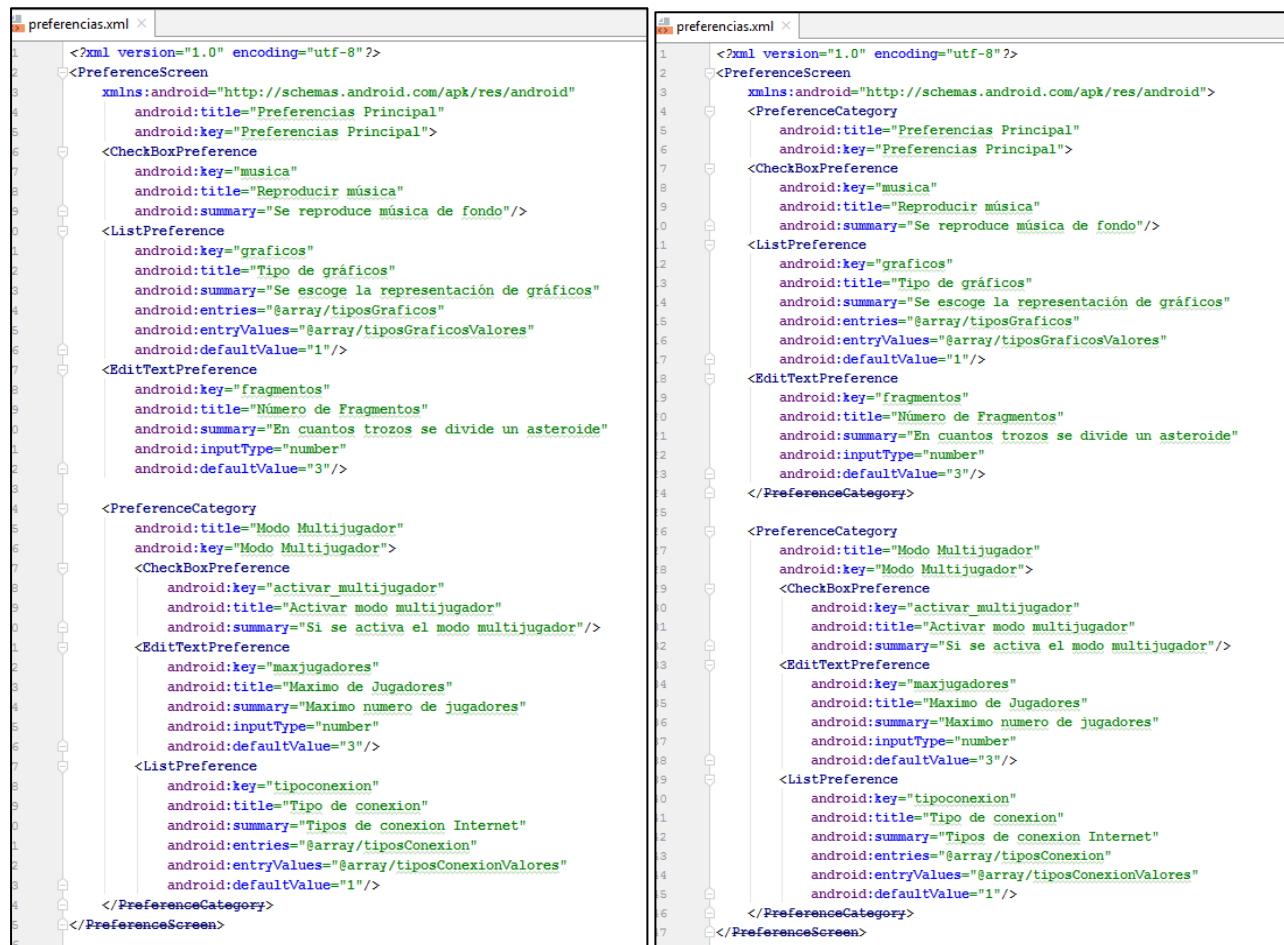
```

A continuación se representa la forma en la que Android muestra las categorías:



Parte IX: Organizando preferencias

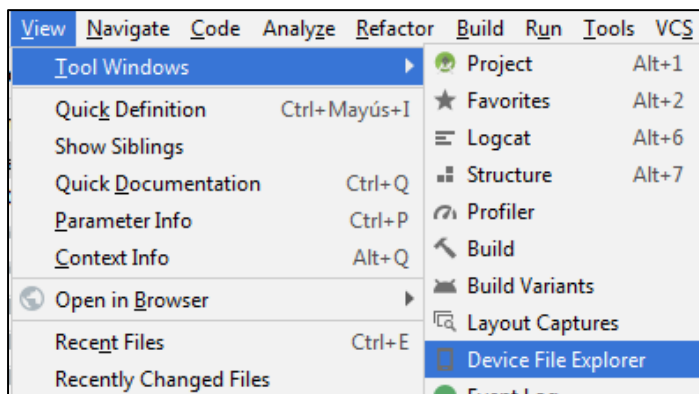
Paso 1. Modifica la práctica anterior para que en lugar de mostrar las propiedades en dos pantallas, las muestre en una sola, tal y como se muestra en la imagen anterior.



Parte X: Donde se almacenan las preferencias de usuario

Veamos donde se han almacenado las preferencias que acabamos de crear:

Paso 1. Para navegar por el sistema de ficheros de un dispositivo con **Android Studio**, abre la opción **View > Tools Windows > Device File Explorer**.



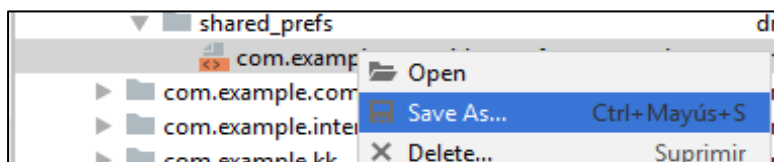
| Name | Permisio... | Date | Size |
|--------------------|-------------|------------------|---------|
| acct | dr-xr-xr-x | 2019-11-07 08:52 | 0 B |
| bin | lrw-r--r-- | 2008-12-31 16:00 | 11 B |
| cache | drwxrwx--- | 2019-10-28 12:47 | 4 KB |
| carrier | drwxr-xr-x | 2008-12-31 16:00 | 4 KB |
| config | drwxr-xr-x | 1970-01-01 01:00 | 0 B |
| d | lrw-r--r-- | 2008-12-31 16:00 | 17 B |
| data | drwxrwx--x | 2019-11-07 08:52 | 4 KB |
| dev | drwxr-xr-x | 2019-11-07 08:52 | 3,7 KB |
| dqmdbg | drwxr-xr-x | 2008-12-31 16:00 | 4 KB |
| efs | drwxrwx--x | 2019-06-28 21:09 | 4 KB |
| etc | lrw-r--r-- | 2008-12-31 16:00 | 11 B |
| keydata | drwxr-xr-x | 2008-12-31 16:00 | 4 KB |
| keyrefuge | drwxr-xr-x | 2008-12-31 16:00 | 4 KB |
| lib | drwxr-xr-x | 2008-12-31 16:00 | 4 KB |
| lost-found | drwx----- | 2008-12-31 16:00 | 16 KB |
| mnt | drwxr-xr-x | 2019-11-07 08:52 | 280 B |
| odm | drwxr-xr-x | 2008-12-31 16:00 | 4 KB |
| oem | drwxr-xr-x | 2008-12-31 16:00 | 4 KB |
| omr | drwxrwx--x | 2019-06-10 23:27 | 4 KB |
| proc | dr-xr-xr-x | 1970-01-01 01:00 | 0 B |
| product | drwxr-xr-x | 2008-12-31 16:00 | 4 KB |
| sbin | drwxr-xr-x | 2008-12-31 16:00 | 4 KB |
| sdcard | lrw-r--r-- | 2008-12-31 16:00 | 21 B |
| storage | drwxr-xr-x | 2019-11-07 08:52 | 100 B |
| sys | dr-xr-xr-x | 2019-11-07 08:52 | 0 B |
| system | drwxr-xr-x | 2008-12-31 16:00 | 4 KB |
| vendor | drwxr-xr-x | 2008-12-31 16:00 | 4 KB |
| audit_filter_table | -rw-r--r-- | 2008-12-31 16:00 | 24,1 KB |
| bugreports | lrw-r--r-- | 2008-12-31 16:00 | 50 B |
| charger | lrw-r--r-- | 2008-12-31 16:00 | 13 B |
| cpefs | lrw-r--r-- | 2008-12-31 16:00 | 17 B |
| default.prop | lrw----- | 2008-12-31 16:00 | 23 B |
| init | -rwxr-x--- | 2008-12-31 16:00 | 7,9 MB |

Paso 2. Busca el siguiente fichero: `/data/data/com.examples.asteroide/shared_prefs/org.examples.asteroide_preferences.xml` (la aplicación tiene que estar siendo ejecutada)

| | | | | | |
|---|--|-----|------------|-------|------------|
| ▶ | com.example.android.softkeyboard | | 2013-01-17 | 14:33 | drwxr-x--x |
| ▲ | com.example.asteroides | | 2013-10-10 | 19:40 | drwxr-x--x |
| ▶ | lib | | 2013-08-26 | 12:01 | drwxr-xr-x |
| ▲ | shared_prefs | | 2013-10-11 | 18:53 | drwxrwx--x |
| | com.example.asteroides_preferences.xml | 260 | 2013-10-11 | 18:53 | -rw-rw---- |

NOTA: En un dispositivo real no tendrás permiso para acceder a estos ficheros (A menos que el dispositivo haya sido rooteado).

Paso 3. Descarga el fichero haciendo click botón derecho "Save As"..



Paso 4. Visualiza su contenido. Tiene que ser similar a:

```

com.example.asteroides_preferences.xml x
1  <?xml version='1.0' encoding='utf-8' standalone='yes' ?>
2  <map>
3      <string name="fragmentos">3</string>
4      <string name="maxjugadores">3</string>
5      <string name="tipoconexion">1</string>
6      <string name="graficos">1</string>
7      <boolean name="musica" value="true" />
8      <boolean name="activar_multijugador" value="true" />
9  </map>

```

Parte XI: Accediendo a los valores de las preferencias

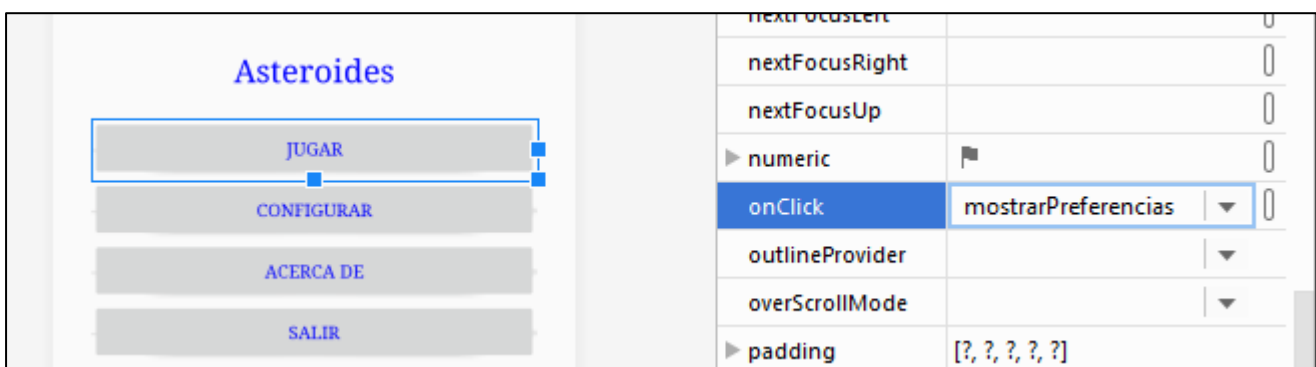
Paso 1. Copia la función anterior en la clase *Asteroides*. Añade el parámetro que se muestra a continuación `mostrarPreferencias(View view)`.

```
import androidx.preference.PreferenceManager;
```

```
public void lanzarPreferencias(View view) {  
    Intent i = new Intent( packageContext: this, Preferencias.class);  
    startActivity(i);  
}  
  
public void mostrarPreferencias(View view) {  
    SharedPreferences pref =  
        PreferenceManager.getDefaultSharedPreferences( context: this);  
    String s = "música: " + pref.getBoolean( s: "musica", b: true)  
        + ", gráficos: " + pref.getString( s: "graficos", s1: "?");  
    Toast.makeText( context: this, s, Toast.LENGTH_SHORT).show();  
}
```

```
dependencies {  
    implementation fileTree(dir: 'libs', include: ['*.jar'])  
    implementation 'androidx.appcompat:appcompat:1.1.0'  
    implementation 'androidx.constraintlayout:constraintlayout:1.1.3'  
    testImplementation 'junit:junit:4.12'  
    androidTestImplementation 'androidx.test:runner:1.2.0'  
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.2.0'  
  
    implementation "androidx.preference:preference:1.1.0"  
}
```

Paso 2. Asígnala al atributo `onClick` del botón *Jugar* el método anterior.



Paso 3. Visualiza también el resto de las preferencias que hayas introducido.



Parte XII: Verificar valores correctos de una preferencia

Paso 1. Copia el siguiente código al final del método onCreate():

```
public class Preferencias extends AppCompatActivity {
    private Context context;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        context=this;
        MyPreferenceFragment mPF = new MyPreferenceFragment();
        getSupportFragmentManager().beginTransaction().
            replace(android.R.id.content, mPF).commit();
        getSupportFragmentManager().executePendingTransactions();

        final EditTextPreference fragmentos = mPF.findPreference( key: "fragmentos");
        fragmentos.setOnPreferenceChangeListener((preference, newValue) -> {
            int valor;
            try {
                valor = Integer.parseInt((String)newValue);
            } catch(Exception e) {
                Toast.makeText(context, text: "Ha de ser un número",
                    Toast.LENGTH_SHORT).show();
                return false;
            }
            if (valor>=0 && valor<=9) {
                fragmentos.setSummary(
                    "En cuantos trozos se divide un asteroide (" +valor+ ")");
                return true;
            } else {
                Toast.makeText(context, text: "Máximo de fragmentos 9",
                    Toast.LENGTH_SHORT).show();
                return false;
            }
        });
    }
}
```

```

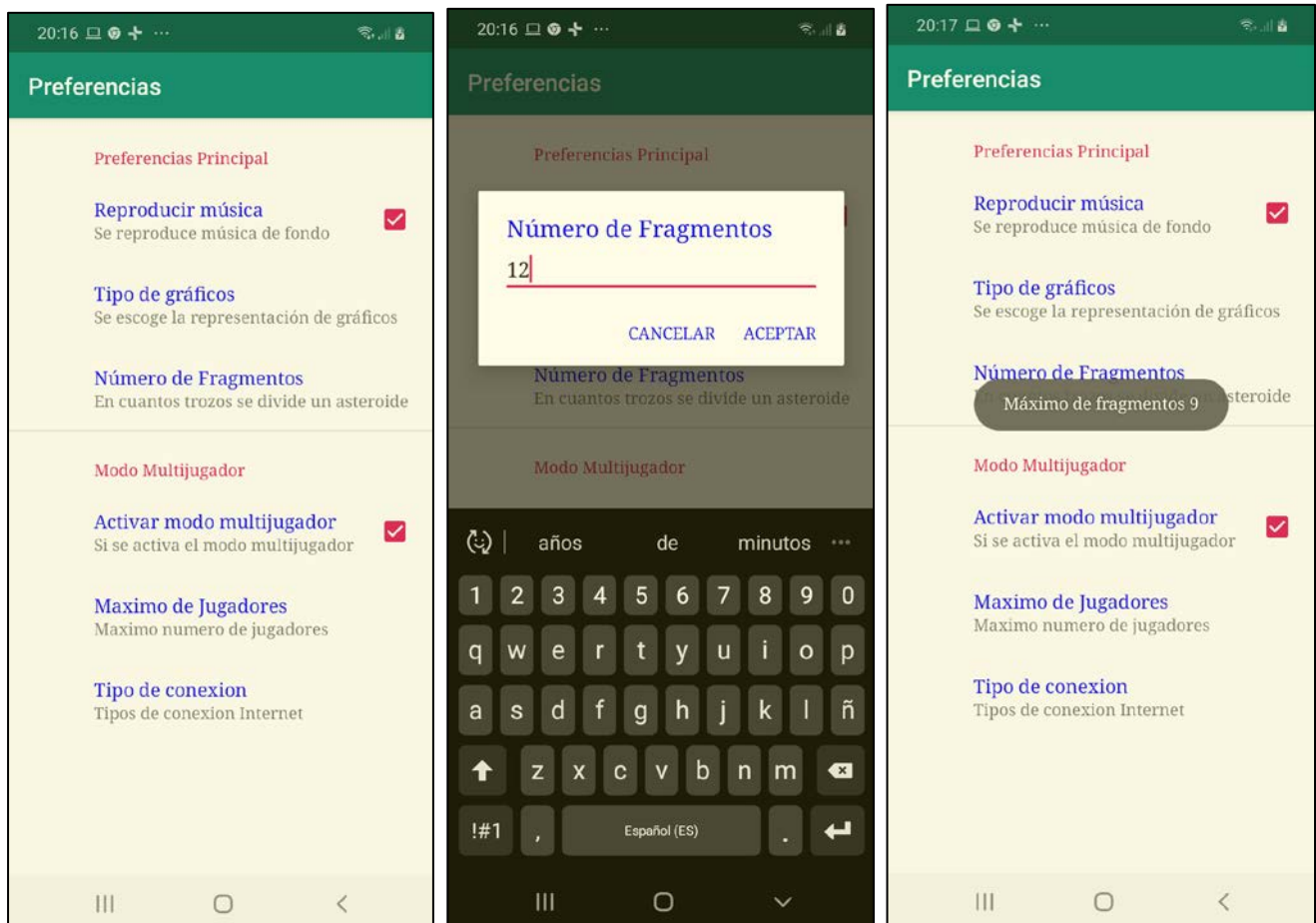
public static class MyPreferenceFragment extends PreferenceFragmentCompat {
    @Override
    public void onCreate(final Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        addPreferencesFromResource(R.xml.preferences);
    }

    @Override
    public void onCreatePreferences(Bundle savedInstanceState, String rootKey) {
    }
}

```

El código comienza obteniendo una referencia de la preferencia fragmentos, para asignarle un escuchador que será llamado cuando cambie su valor. El escuchador comienza convirtiendo el valor introducido a entero. En caso de producirse un error es porque el usuario no ha introducido un valor adecuado. En este caso, mostramos un mensaje y devolvemos false para que el valor de la preferencia no sea modificado. Si no hay error, tras verificar el rango de valores aceptables, modificamos la explicación de la preferencia para que aparezca el nuevo valor entre paréntesis y devolvemos true para aceptar este valor. Si no está en el rango, mostramos un mensaje indicando el problema y devolvemos false.

Paso 2. Ejecuta el proyecto y verifica que funciona correctamente.



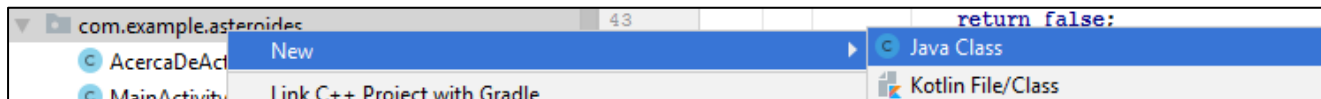
Paso 3. Mostrar el valor de una preferencia

En el ejercicio anterior cuando se modifica el número de fragmentos se muestra entre paréntesis el nuevo valor introducido. El funcionamiento no es del todo correcto, cuando entramos por primera vez o cuando se cambia el teléfono de vertical a horizontal este valor no se muestra. Añade el código necesario para que este valor aparezca siempre.

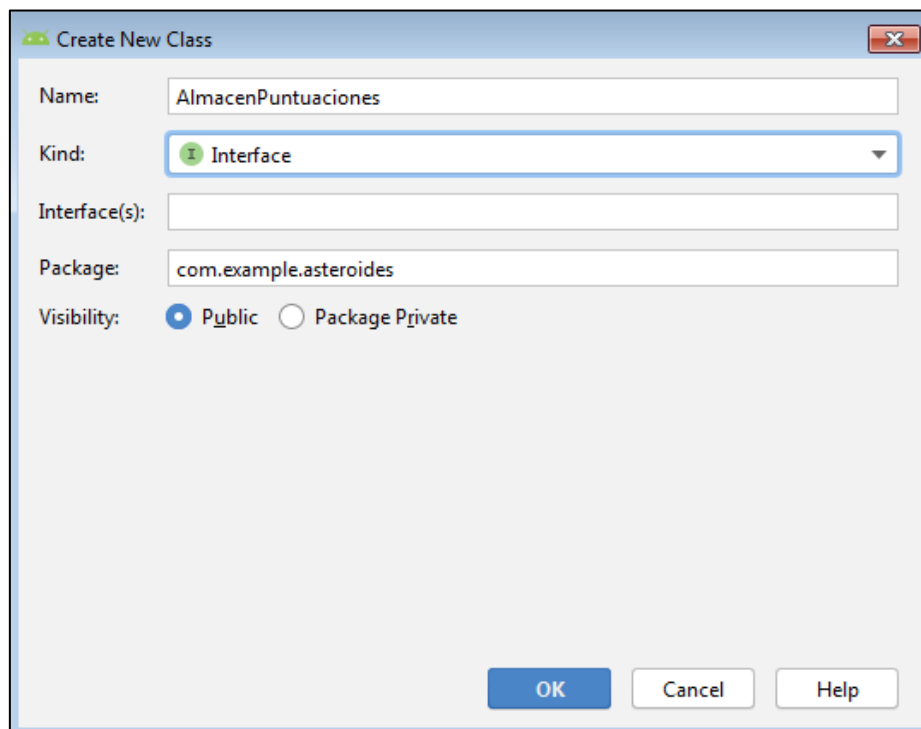
Parte XIII: El interfaz AlmacenPuntuaciones

Paso 1. Abre la aplicación Asteroides.

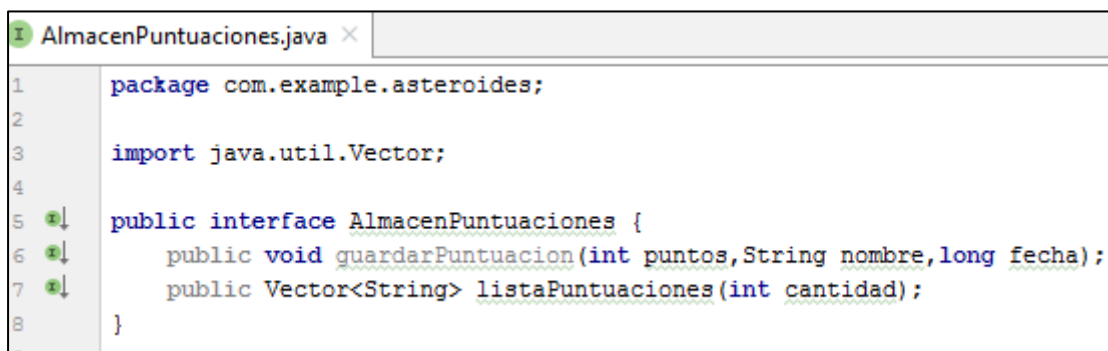
Paso 2. Pulsa con el botón derecho sobre la carpeta de código (*com.example.asteroides*) y selecciona *New > Java Class*.



Paso 3. En el campo *Name*: introduce *AlmacenPuntuaciones*, en el campo *Kind* introduce *Interface* y pulsa *Ok*.



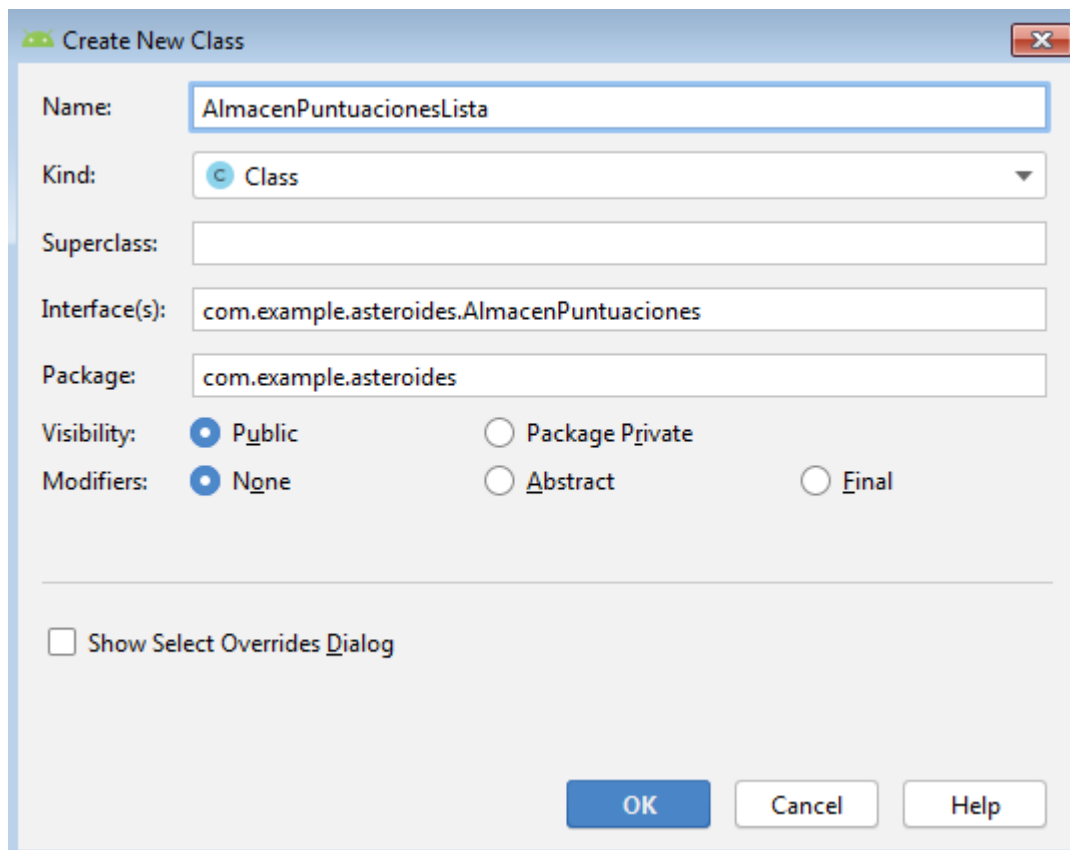
Paso 4. Introduce el código que se muestra a continuación:



Nota sobre Java: La interfaz es una clase abstracta pura, es decir una clase donde se indican los métodos pero no se implementa ninguno (en este caso se dice que los métodos son abstractos). Permite al programador de la clase establecer la estructura de esta (nombres de métodos, sus parámetros y tipos que retorna, pero no el código de cada método). Una interfaz también puede contener constantes, es decir campos de tipo static y final.

Las diferentes clases que definamos para almacenar puntuaciones han de implementar esta interfaz. Como ves tiene dos métodos. El primero para guardar la puntuación de una partida, con los parámetros puntuación obtenida, nombre del jugador y fecha de la partida. La segunda es para obtener una lista de puntuaciones previamente almacenadas. El parámetro *cantidad* indica el número máximo de puntuaciones que ha de devolver.

Paso 5. Veamos a continuación una clase que utiliza esta interfaz. Para ello crea en el proyecto la clase `AlmacenPuntuacionesLista`.



Create New Class

Name:

Kind: ☒ Class

Superclass:

Interface(s):

Package:

Visibility: ☒ Public ☐ Package Private

Modifiers: ☒ None ☐ Abstract ☐ Final

☐ Show Select Overrides Dialog

OK Cancel Help

Paso 6. Introduce el siguiente código:



```
package com.example.asteroides;

import java.util.Vector;

public class AlmacenPuntuacionesLista implements AlmacenPuntuaciones {

    private Vector puntuaciones;

    public AlmacenPuntuacionesLista() {
        puntuaciones= new Vector();
        puntuaciones.add("123000 Pepito Dominguez");
        puntuaciones.add("111000 Pedro Martinez");
        puntuaciones.add("011000 Paco Pérez");
    }

    @Override
    public void guardarPuntuacion(int puntos, String nombre, long fecha) {
        puntuaciones.add( index: 0, element: puntos + " " + nombre);
    }

    @Override
    public Vector listaPuntuaciones(int cantidad) {
        return puntuaciones;
    }

}
```

Esta clase almacena la lista de puntuaciones en un vector de String. Tiene el inconveniente de que al tratarse de una variable local, cada vez que se cierre la aplicación se perderán las puntuaciones. El

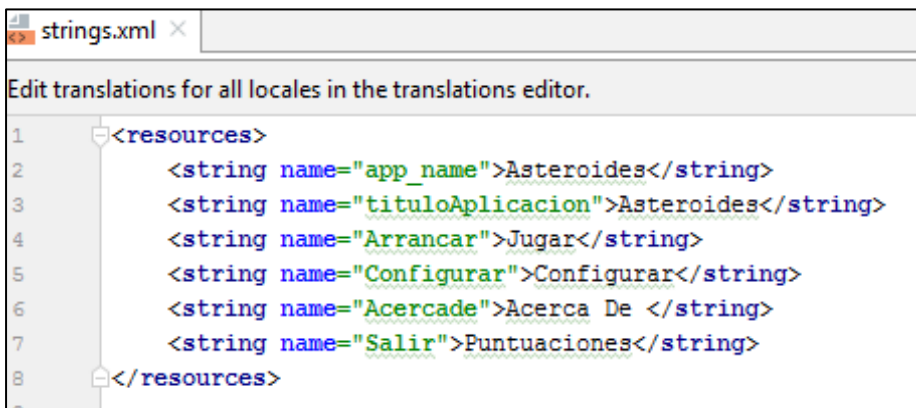
constructor inicializa el array e introduce tres valores. La idea es que aunque todavía no esté programado el juego y no podamos jugar, tengamos ya algunas puntuaciones para poder representar una lista. El método `guardarPuntuacion()` se limita a insertar en la primera posición del array un String con los puntos y el nombre. La fecha no es almacenada. El método `listaPuntuaciones()` devuelve el vector de String entero, sin tener en cuenta el parámetro `cantidad` que debería limitar el número de Strings devueltos.

Paso 7. En la actividad MainActivity tendrás que declarar una variable para almacenar las puntuaciones:

```
public class MainActivity extends AppCompatActivity {  
  
    public static AlmacenPuntuaciones almacen= new AlmacenPuntuacionesLista();  
  
    private Button bAcercaDe;  
    private Button bSalir;
```

Nota sobre Java: El modificador `static` permite compartir el valor de una variable entre todos los objetos de la clase. Es decir, aunque se creen varios objetos, solo existirá una única variable `almacen` compartida por todos los objetos. El modificador `public` permite acceder a la variable desde fuera de la clase. Por lo tanto, no será necesario crear métodos getters y setters. Para acceder a esta variable no tendremos más que escribir el nombre de la clase seguida de un punto y el nombre de la variable. Es decir `MainActivity.almacen`.

Paso 8. Para que los jugadores puedan ver las últimas puntuaciones obtenidas, modifica el cuarto botón del `layout activity_main.xml` para que en lugar del texto “Salir” se visualice “Puntuaciones”. Para ello modifica los ficheros `res/values/strings`. También sería interesante que cambiaras el fichero `res/values-en/strings`.



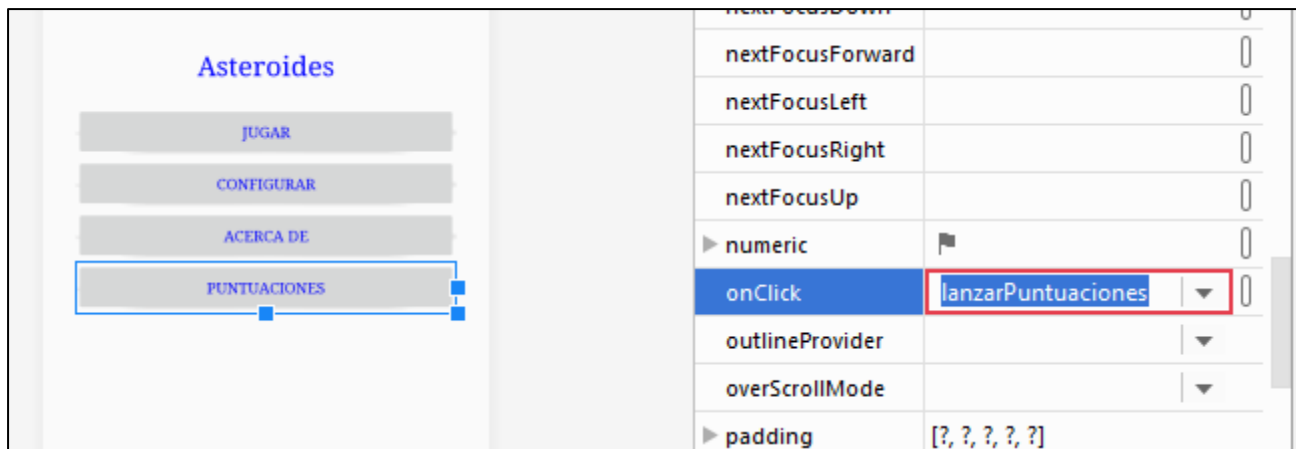
```
<?xml version="1.0" encoding="utf-8"?>  
<resources>  
    <string name="app_name">Asteroides</string>  
    <string name="tituloAplicacion">Asteroides</string>  
    <string name="Arrancar">Jugar</string>  
    <string name="Configurar">Configurar</string>  
    <string name="Acercade">Acerca De </string>  
    <string name="Salir">Puntuaciones</string>  
</resources>
```



```
<?xml version="1.0" encoding="utf-8"?>  
<resources>  
    <string name="app_name">Asteroids</string>  
    <string name="tituloAplicacion">Asteroids</string>  
    <string name="Arrancar">Play</string>  
    <string name="Configurar">Configuration</string>  
    <string name="Acercade">About </string>  
    <string name="Salir">Scores</string>  
</resources>
```

Paso 9. Modifica el escuchador asociado al cuarto botón para que llame al método `lanzarPuntuaciones`:

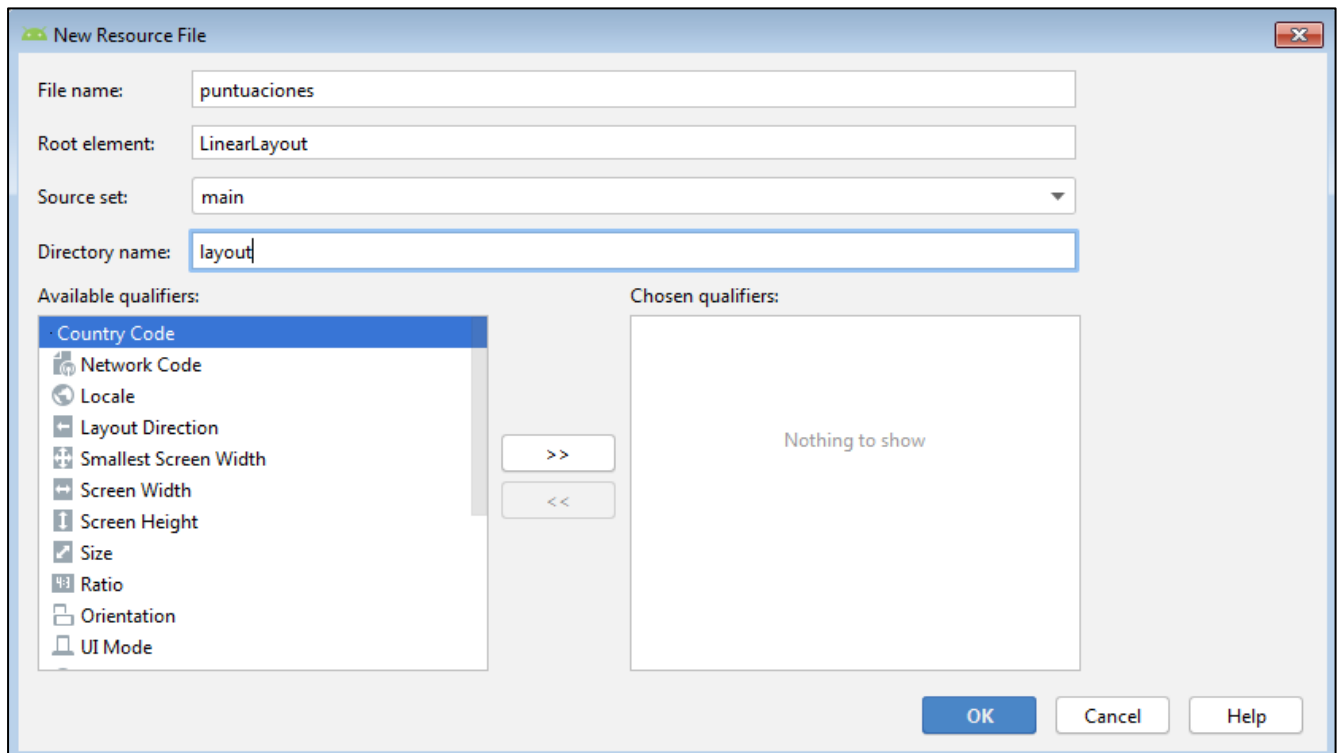
```
public void lanzarPuntuaciones(View view){
    Intent i = new Intent( packageContext: this, Puntuaciones.class);
    startActivity(i);
}
```



Paso 10. De momento no te permitirá ejecutar la aplicación. Hasta que en el siguiente apartado no creamos la actividad **Puntuaciones** no será posible.

Parte XIV: Un ListView que visualiza una lista de Strings (I)

Paso 1. El *Layout* que utilizaremos en *Asteroides* para mostrar las puntuaciones se llamará *puntuaciones.xml*. En él se incluye una vista *ListView*. Crea el *Layout* con el siguiente código:




```
puntuaciones.xml x
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     android:orientation="vertical"
4     android:layout_width="match_parent"
5     android:layout_height="match_parent">
6     <TextView
7         android:layout_width="match_parent"
8         android:layout_height="wrap_content"
9         android:text="Puntuaciones"
10        android:gravity="center"
11        android:layout_margin="10px"
12        android:textSize="10pt"/>
13    <FrameLayout
14        android:layout_width="match_parent"
15        android:layout_height="0dip"
16        android:layout_weight="1">
17        <ListView
18            android:id="@android:id/list"
19            android:layout_width="match_parent"
20            android:layout_height="match_parent"
21            android:drawSelectorOnTop="false" />
22        <TextView
23            android:id="@android:id/empty"
24            android:layout_width="match_parent"
25            android:layout_height="match_parent"
26            android:text="No hay puntuaciones" />
27    </FrameLayout>
28 </LinearLayout>
```

Paso 2. Necesitamos ahora crear la actividad **Puntuaciones** para visualizar el Layout anterior. Crea una nueva clase en tu proyecto e introduce el siguiente código:

```
Puntuaciones.java x
1 package com.example.asteroides;
2
3 import android.app.ListActivity;
4 import android.os.Bundle;
5 import android.widget.ArrayAdapter;
6
7 public class Puntuaciones extends ListActivity {
8
9     @Override
10    protected void onCreate(Bundle savedInstanceState) {
11        super.onCreate(savedInstanceState);
12        setContentView(R.layout.puntuaciones);
13        setListAdapter(new ArrayAdapter< context: this,
14            android.R.layout.simple_list_item_1,
15            MainActivity.almacen.listaPuntuaciones( cantidad: 10) >());
16    }
17 }
18 }
```

Toda actividad que vaya a visualizar un ListView ha de heredar de ListActivity. Además, ha de llamar al método setListAdapter() para indicar el adaptador con la lista de elementos a visualizar. En el

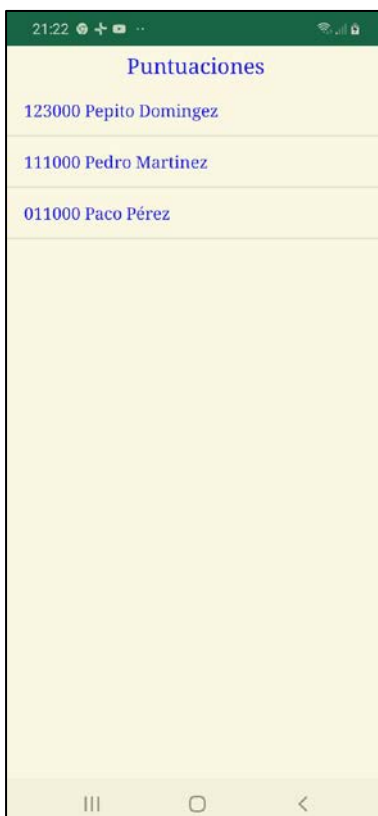
ejemplo se ha utilizado una de las posibilidades más sencillas, para crear un adaptador, usar la clase `ArrayAdapter<clase>`. Un `ArrayAdapter` crea las vistas del `ListView` a partir de los datos almacenados en un array. Puedes utilizar un array que contenga datos de cualquier clase, no tienes más que indicar en `<Clase>` la clase deseada. En este caso se utiliza de un array de `String[1]`. El constructor de `ArrayAdapter<clase>` tiene tres parámetros: El primer parámetro es un `Context` con información sobre el entorno de la aplicación. Utilizaremos como contexto la misma actividad que hace la llamada. El segundo parámetro es un `Layout`, utilizado para representar cada elemento de la lista. En este ejemplo, en lugar de definir uno nuevo, utilizaremos una ya definido en el sistema. El último parámetro es un array con los strings a mostrar. Para ello, llamamos al método `listaPuntuaciones()` que nos devuelve esta lista del objeto estático almacen de la clase `MainActivity`.

Paso 3. Recuerda que toda nueva actividad ha de ser registrada en `AndroidManifest.xml`.

```
<activity android:name=".MainActivity">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
<activity
    android:name=".AcercaDeActividad"
    android:label="Acerca de ..."
    android:theme="@style/Theme.AppCompat.Light.Dialog" />
<activity
    android:name=".Preferencias"
    android:label="Preferencias" />
<activity android:name=".Puntuaciones"></activity>
```

Paso 4. Prueba si funcionan las modificaciones introducidas.



Parte XV: Un ListView que visualiza layouts personalizados

Paso 1. Reemplaza la clase anterior por:

```
public class Puntuaciones extends ListActivity {  
  
    @Override public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.puntuaciones);  
        setListAdapter(  
            new ArrayAdapter(this,  
                R.layout.elemento_lista,  
                R.id.titulo,  
                MainActivity.almacen.listaPuntuaciones( cantidad: 10)));  
    }  
}
```

Como hemos explicado, la clase `ArrayAdapter<String>` permite insertar los datos desde un array de `String` en nuestro `ListView`. En este ejemplo se utiliza un constructor con cuatro parámetros:

- **this**: es el contexto, con información sobre el entorno de la aplicación.
- **R.layout.elemento_lista**: es una referencia de recurso a la vista que será utilizada repetidas veces para formar la lista. Se define a continuación.
- **R.id.titulo**: identifica un id de la vista anterior que ha de ser un `TextView`. Su texto será reemplazado por el que se indica en el siguiente parámetro.
- **MainActivity.almacen.listaPuntuaciones(10)**: vector de `String` con los textos que serán visualizados en cada uno de los `TextView`. Esta lista es obtenida accediendo a la clase `Asteroides` a su variable estática `almacen` llamando a su método `listaPuntuaciones()`.

Paso 2. Ahora hemos de definir el Layout que representará cada uno de los elementos de la lista. Crea el fichero `res/Layout/elemento_lista.xml` con el siguiente código:

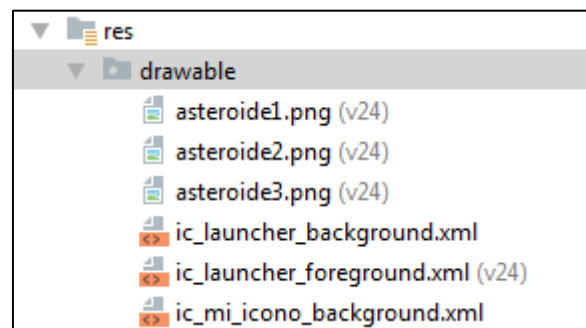
```

elemento_lista.xml
1  <?xml version="1.0" encoding="utf-8" ?>
2  <RelativeLayout
3      xmlns:android="http://schemas.android.com/apk/res/android"
4      android:layout_width="match_parent"
5      android:layout_height="?android:attr/listPreferredItemHeight">
6      <ImageView android:id="@+id/icono"
7          android:layout_width="?android:attr/listPreferredItemHeight"
8          android:layout_height="match_parent"
9          android:layout_alignParentLeft="true"
10         android:src="@drawable/asteroide2"/>
11      <TextView android:id="@+id/titulo"
12          android:layout_width="match_parent"
13          android:layout_height="wrap_content"
14          android:layout_toRightOf="@id/icono"
15          android:layout_alignParentTop="true"
16          android:textAppearance="?android:attr/textAppearanceLarge"
17          android:singleLine="true" />
18      <TextView android:id="@+id/subtitulo"
19          android:layout_width="match_parent"
20          android:layout_height="match_parent"
21          android:text="Otro Texto"
22          android:layout_toRightOf="@id/icono"
23          android:layout_below="@id/titulo"
24          android:layout_alignParentBottom="true"
25          android:gravity="center"/>
26  </RelativeLayout>

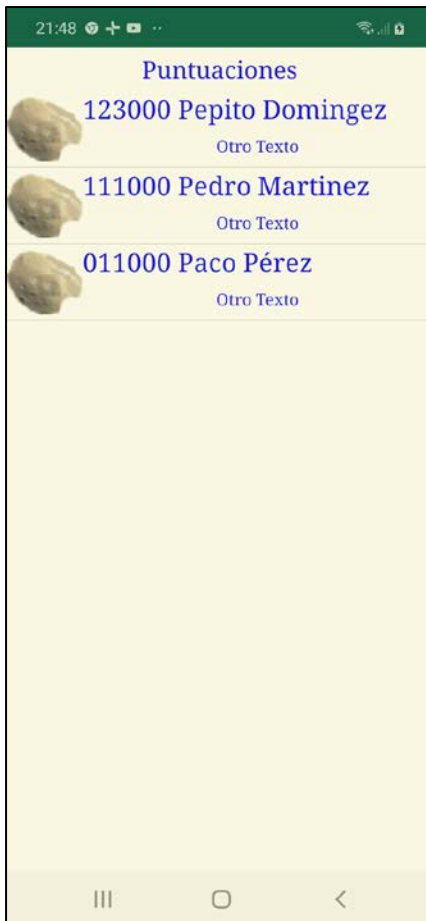
```

Este Layout representa una imagen a la izquierda con dos textos a la derecha, uno de mayor tamaño en la parte superior. Para combinar estos elementos se ha escogido un **RelativeLayout**, donde el ancho se establece a partir de un parámetro de configuración del sistema **?android:attr/listPreferredItemHeight**. El primer elemento que contiene es un **ImageView** alineado a la izquierda. Su alto es la misma que el contenedor (**match_parent**) mientras que el ancho se establece con el mismo parámetro que el alto del contenedor. Por lo tanto la imagen será cuadrada.

Paso 3. Puedes descargar las imágenes utilizadas en la aplicación Asteroides de www.androidcurso.com. En el menú El gran libro de Android / Ficheros usados en ejercicios dentro de Graficos.zip. Copia el fichero **asteriode1.png**, **asteriode2.png** y **asteriode3.png** a la carpeta **res/drawable**.

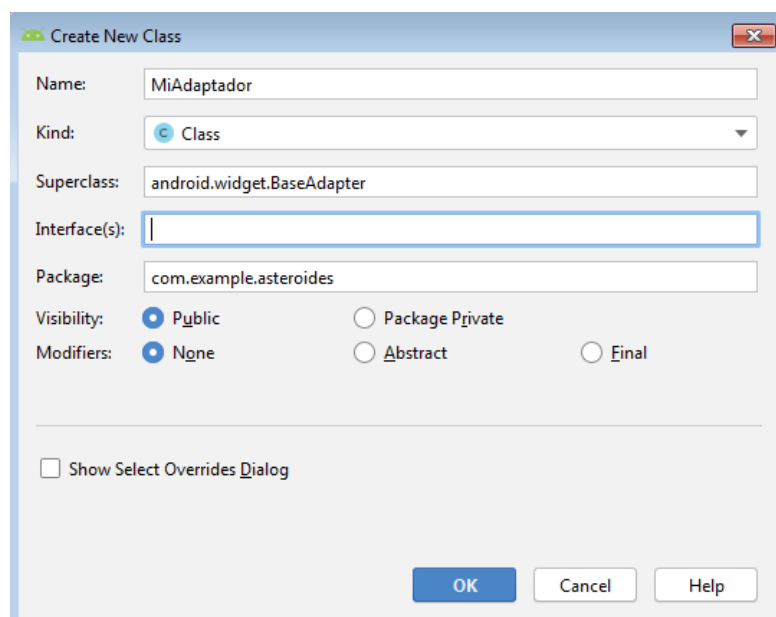


Paso 4. Ejecuta la aplicación y verifica el resultado.



Parte XVI: Un ListView con nuestro propio adaptador

Paso 1. Crea la clase *MiAdaptador.java* en el proyecto con el siguiente código:



```

public class MiAdaptador extends BaseAdapter {
    private final Activity actividad;
    private final Vector lista;
    public MiAdaptador(Activity actividad, Vector lista) {
        super();
        this.actividad = actividad;
        this.lista = lista;
    }
    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        LayoutInflater inflater = actividad.getLayoutInflater();
        View view = inflater.inflate(R.layout.elemento_lista, root: null, attachToRoot: true);
        TextView textView = (TextView)view.findViewById(R.id.titulo);
        textView.setText(lista.elementAt(position).toString());
        ImageView imageView=(ImageView)view.findViewById(R.id.icono);
        switch (Math.round((float)Math.random()*3)){
            case 0:
                imageView.setImageResource(R.drawable.asteroide1);
                break;
            case 1:
                imageView.setImageResource(R.drawable.asteroide2);
                break;
            default:
                imageView.setImageResource(R.drawable.asteroide3);
                break;
        }
        return view;
    }
    @Override
    public int getCount() {
        return lista.size();
    }
    @Override
    public Object getItem(int arg0) {
        return lista.elementAt(arg0);
    }
    @Override
    public long getItemId(int position) {
        return position;
    }
}

```

En el constructor de la clase se indica la actividad donde se ejecutará y la lista de datos a visualizar. El método más importante de esta clase es `getView()` el cual tiene que construir los diferentes *Layouts* que serán añadidos en la lista. Comenzamos construyendo un objeto *View* a partir del código xml definido en *elemento_lista.xml*. Este trabajo se realiza por medio de la clase *LayoutInflater*. Luego, se modifica el texto de uno de los *TextView* según el array que se pasó en el constructor. Finalmente, se obtiene un número al azar (`Math.round()`) y se asigna uno de los tres gráficos de forma aleatoria.

Paso 2. Reemplaza en la clase *Puntuaciones* la llamada al constructor de *ArrayAdapter<String>* por:

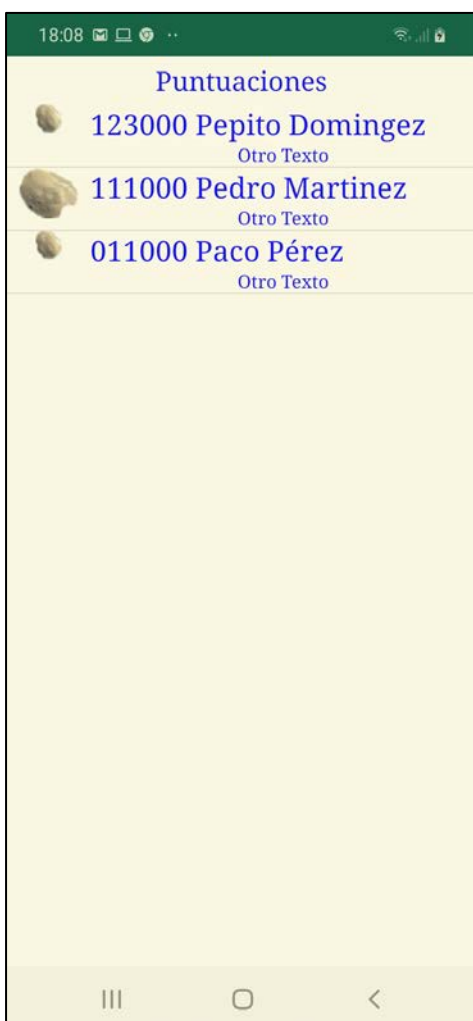
```

public class Puntuaciones extends ListActivity {

    @Override public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.puntuaciones);
        //setListAdapter( new ArrayAdapter(this,
        //                                R.layout.elemento_lista,
        //                                R.id.titulo,
        //                                MainActivity.almacen.listaPuntuaciones(10)));
        setListAdapter(new MiAdaptador( actividad: this,
                                        MainActivity.almacen.listaPuntuaciones( cantidad: 10)));
    }
}

```

Paso 3. Ejecuta la aplicación y verifica el resultado.



Parte XVII: Detectar una pulsación sobre un elemento de la lista

Paso 1. Añade el siguiente método a la clase *Puntuaciones.java*:

```
@Override
protected void onListItemClick(ListView listView,
                                View view, int position, long id) {
    super.onListItemClick(listView, view, position, id);
    Object o = getListAdapter().getItem(position);
    Toast.makeText( context: this, text: "Selección: " + Integer.toString(position)
                  + " - " + o.toString(), Toast.LENGTH_LONG).show();
}
```

Paso 2. Ejecuta la aplicación, pulsa en “Puntuaciones” y luego en una puntuación para verificar el resultado.

