

## PRACTICA 13: ASTEROIDES

### 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>

```

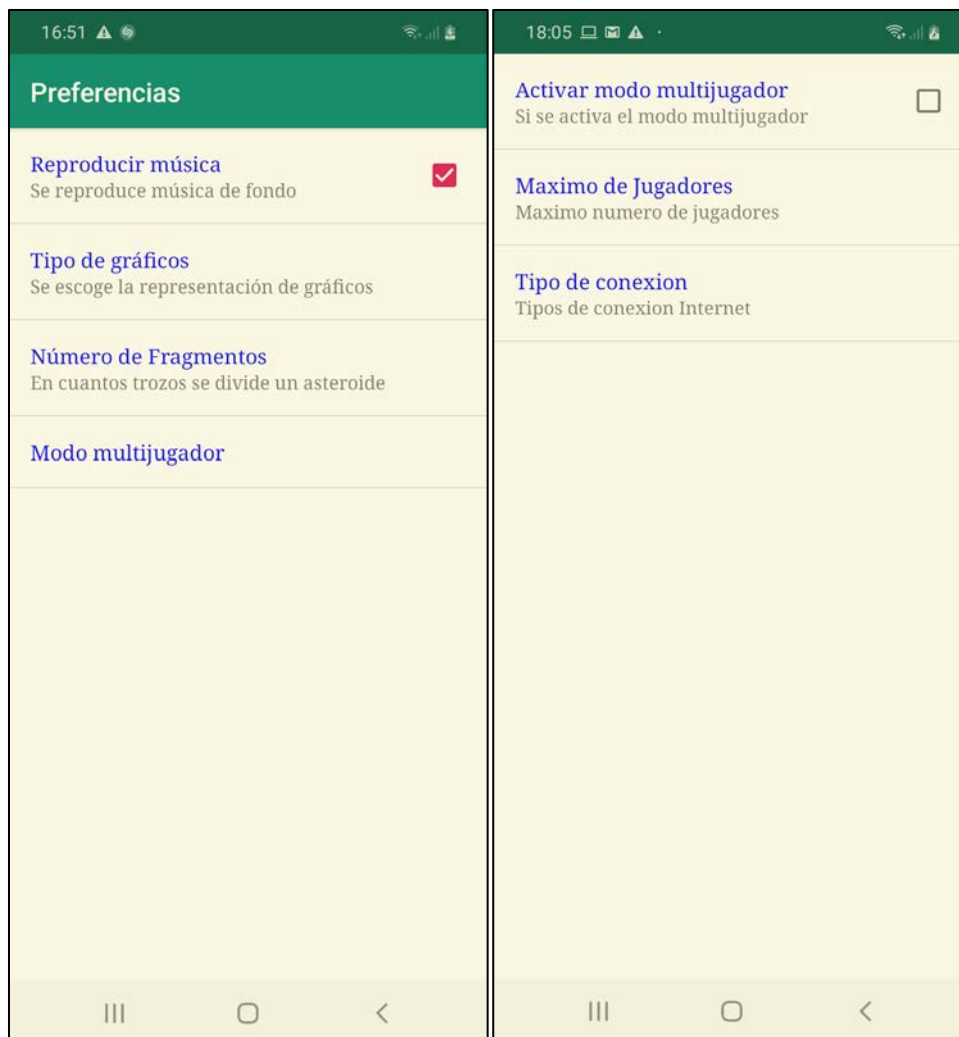
```

<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.



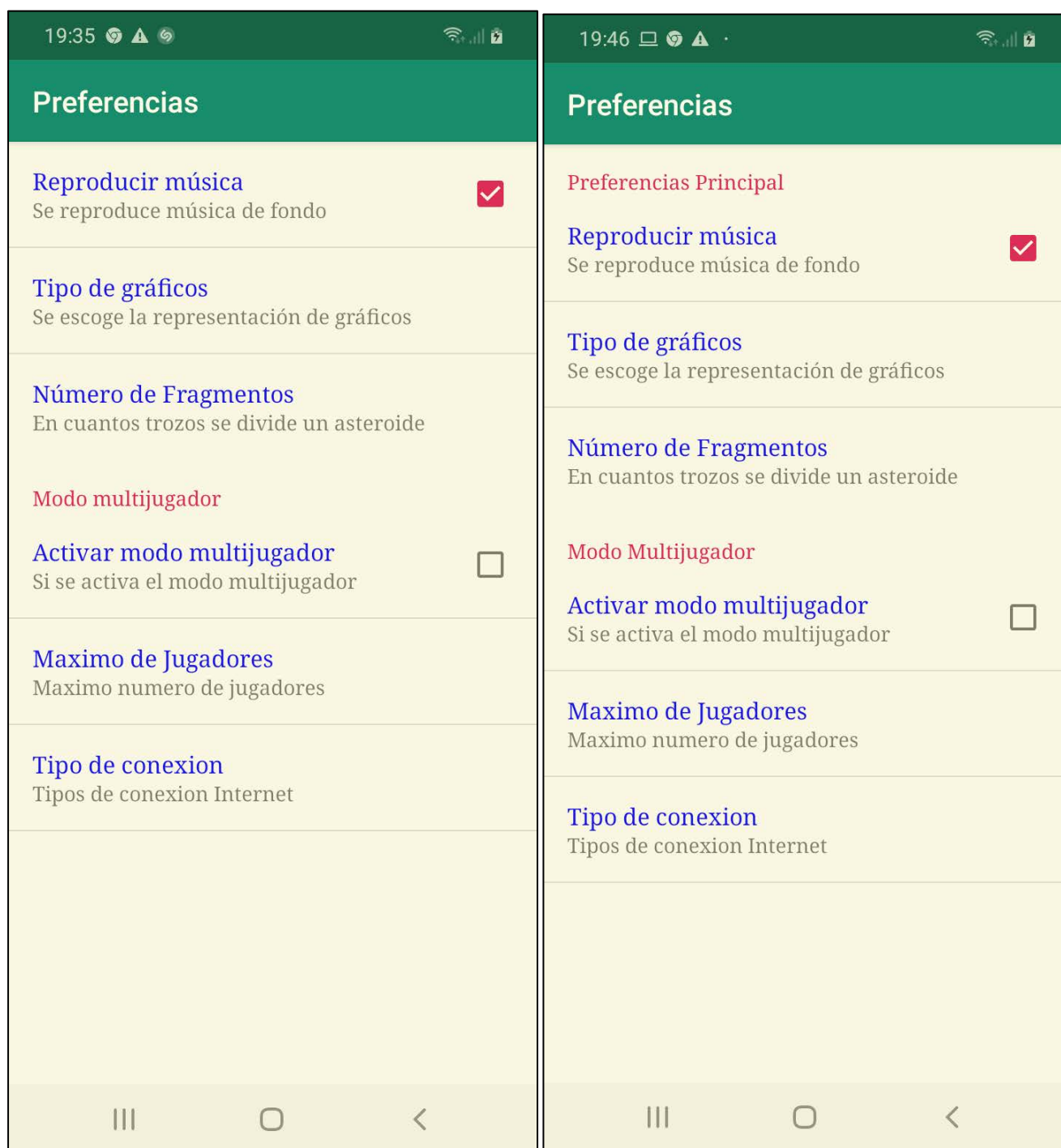
**Paso 5.** Otra opción para organizar las preferencias consiste en agruparlas por categorías. Con esta opción se visualizarán en la misma pantalla, pero separadas por grupos. Has de seguir el siguiente esquema:

```

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

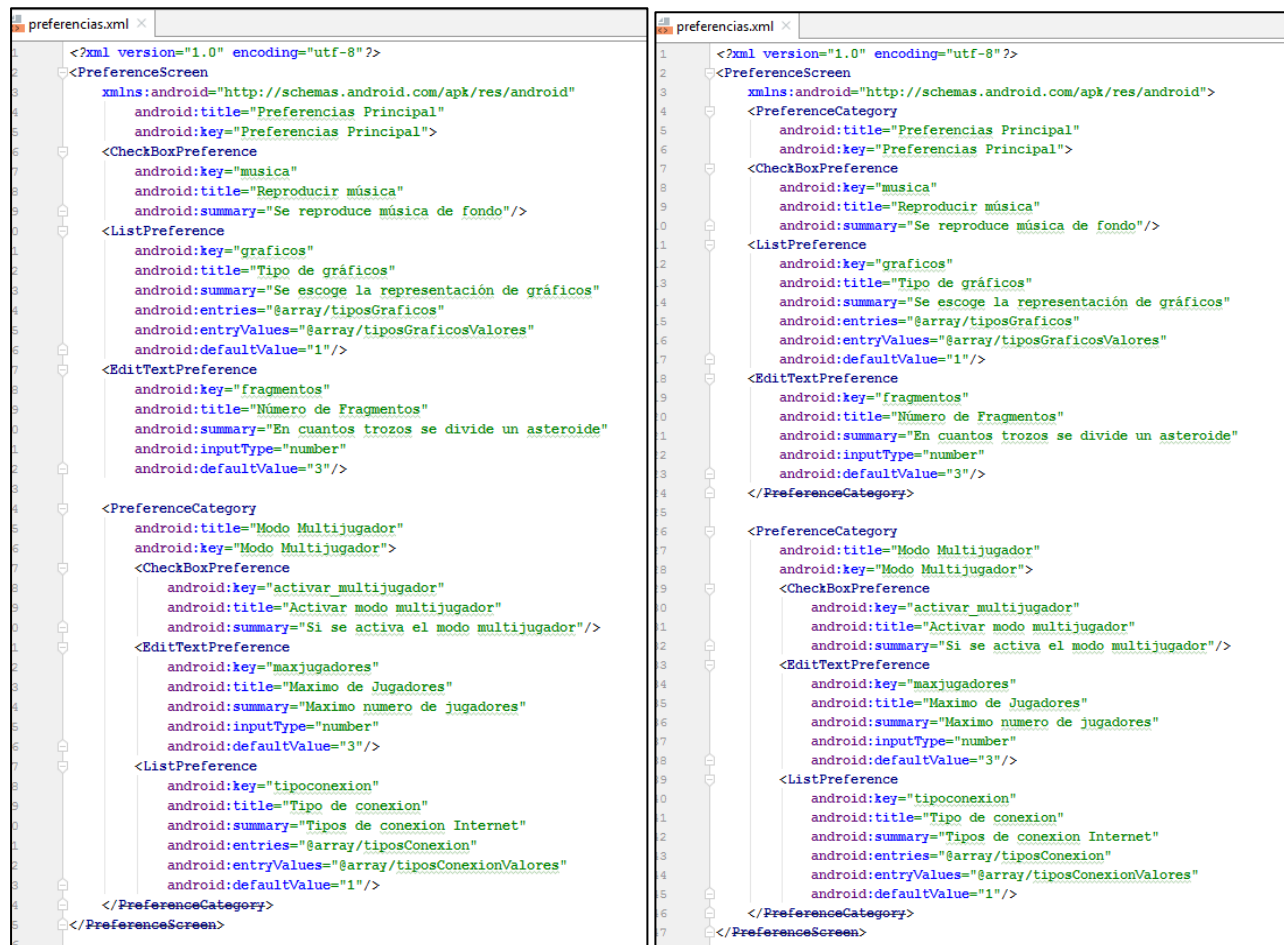
```

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



## Parte IX: Organizando preferencias

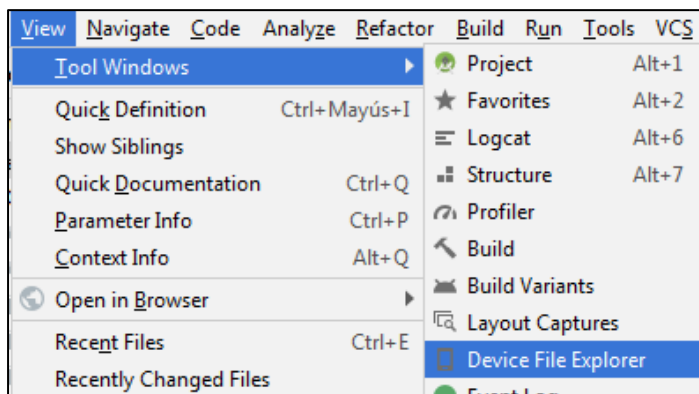
**Paso 1.** Modifica la práctica anterior para que en lugar de mostrar las propiedades en dos pantallas, las muestre en una sola, tal y como se muestra en la imagen anterior.



## 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**.



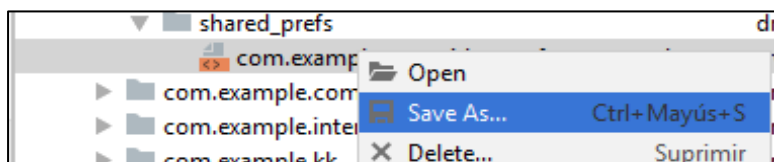
Name	Permisio...	Date	Size
acct	dr-xr-xr-x	2019-11-07 08:52	0 B
bin	lrw-r--r--	2008-12-31 16:00	11 B
cache	drwxrwx---	2019-10-28 12:47	4 KB
carrier	drwxr-xr-x	2008-12-31 16:00	4 KB
config	drwxr-xr-x	1970-01-01 01:00	0 B
d	lrw-r--r--	2008-12-31 16:00	17 B
data	drwxrwx--x	2019-11-07 08:52	4 KB
dev	drwxr-xr-x	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-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;

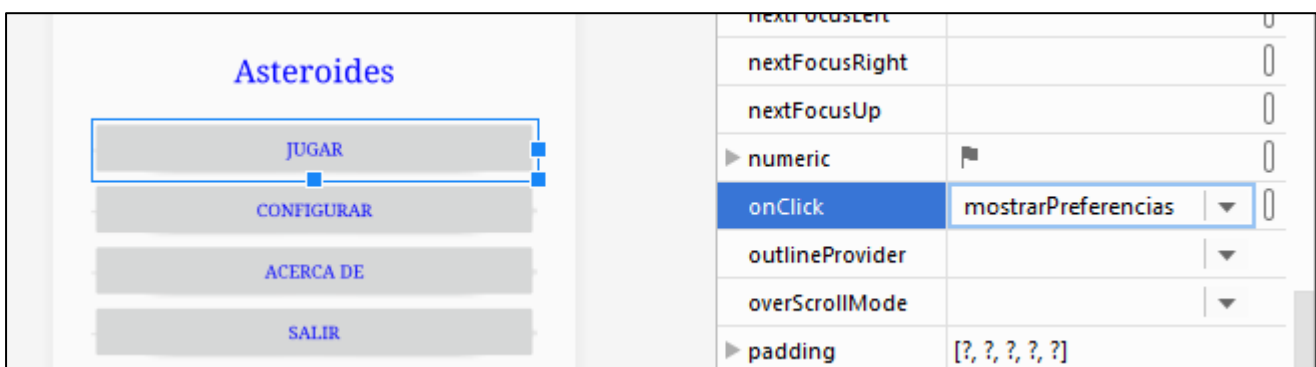
public void lanzarPreferencias(View view) {
    Intent i = new Intent( packageContext: this, Preferencias.class);
    startActivity(i);
}

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;
            }
        });
    }
}
```

```

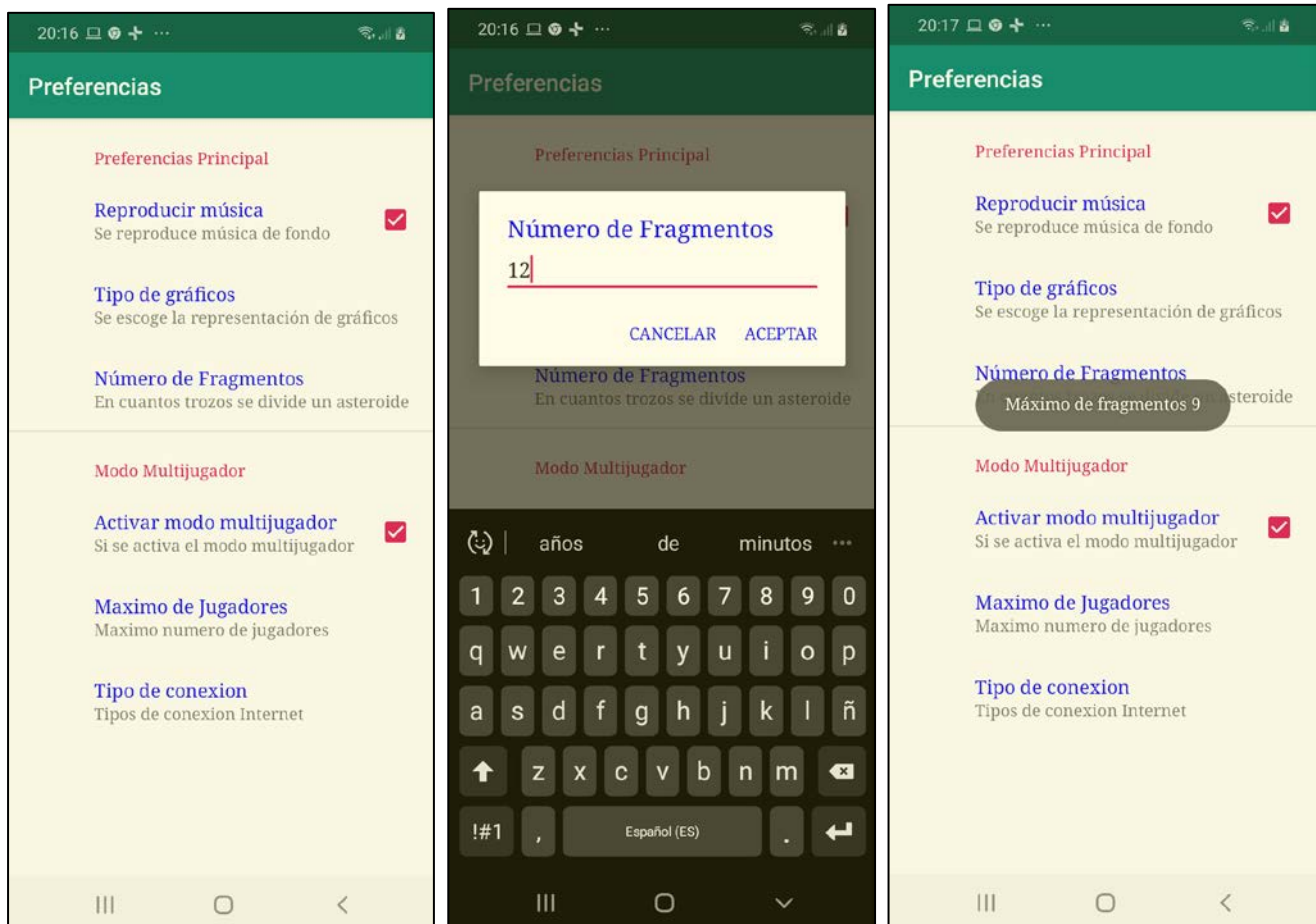
public static class MyPreferenceFragment extends PreferenceFragmentCompat {
    @Override
    public void onCreate(final Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        addPreferencesFromResource(R.xml.preferences);
    }

    @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**

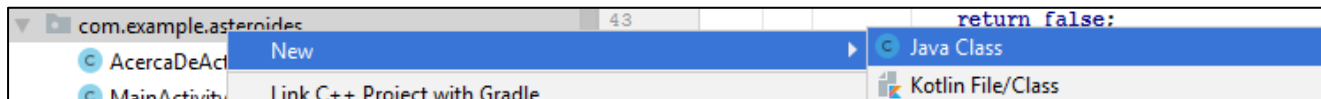
En el ejercicio anterior cuando se modifica el número de fragmentos se muestra entre paréntesis el nuevo valor introducido. El funcionamiento no es del todo correcto, cuando entramos por primera vez o cuando se cambia el teléfono de vertical a horizontal este valor no se muestra. Añade el código necesario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



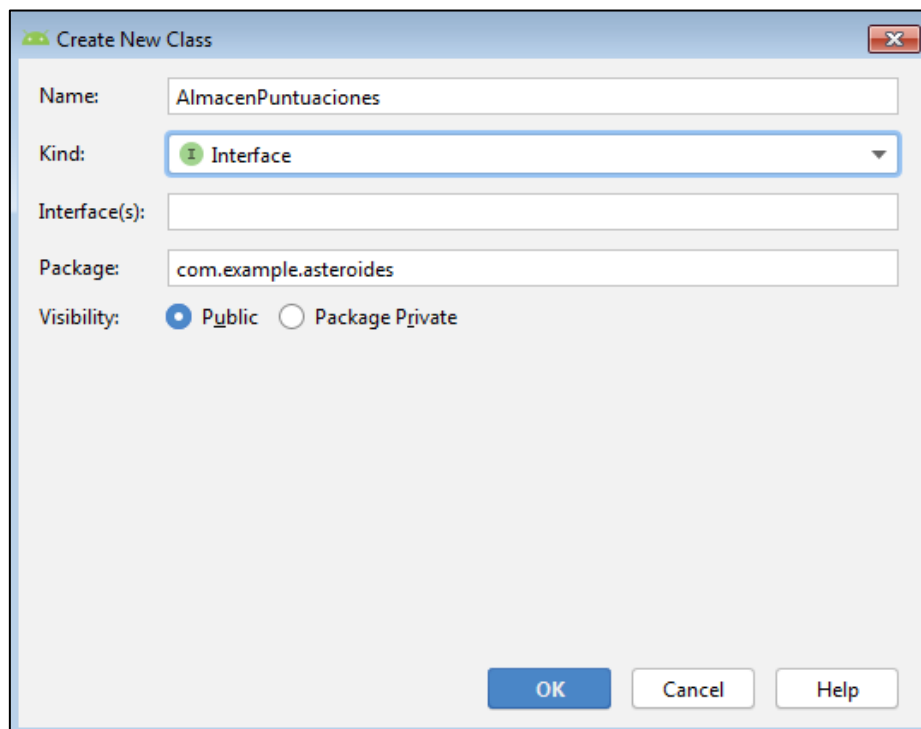
### Parte XIII: El interfaz AlmacenPuntuaciones

**Paso 1.** Abre la aplicación Asteroides.

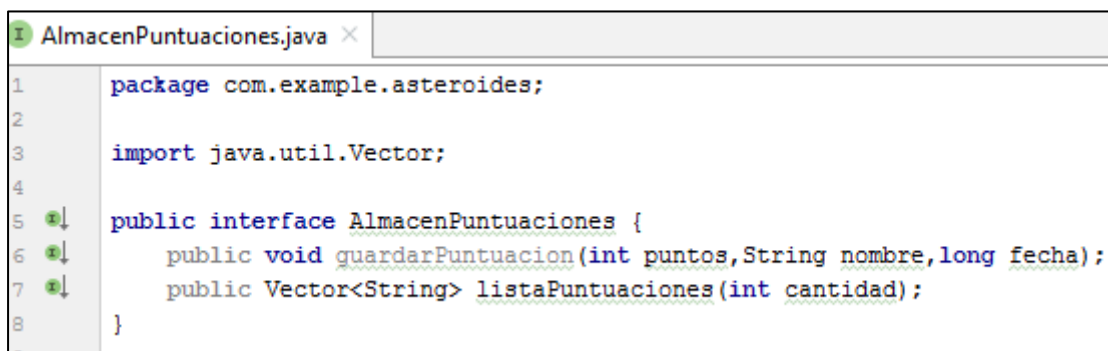
**Paso 2.** Pulsa con el botón derecho sobre la carpeta de código (*com.example.asteroides*) y selecciona *New > Java Class*.



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



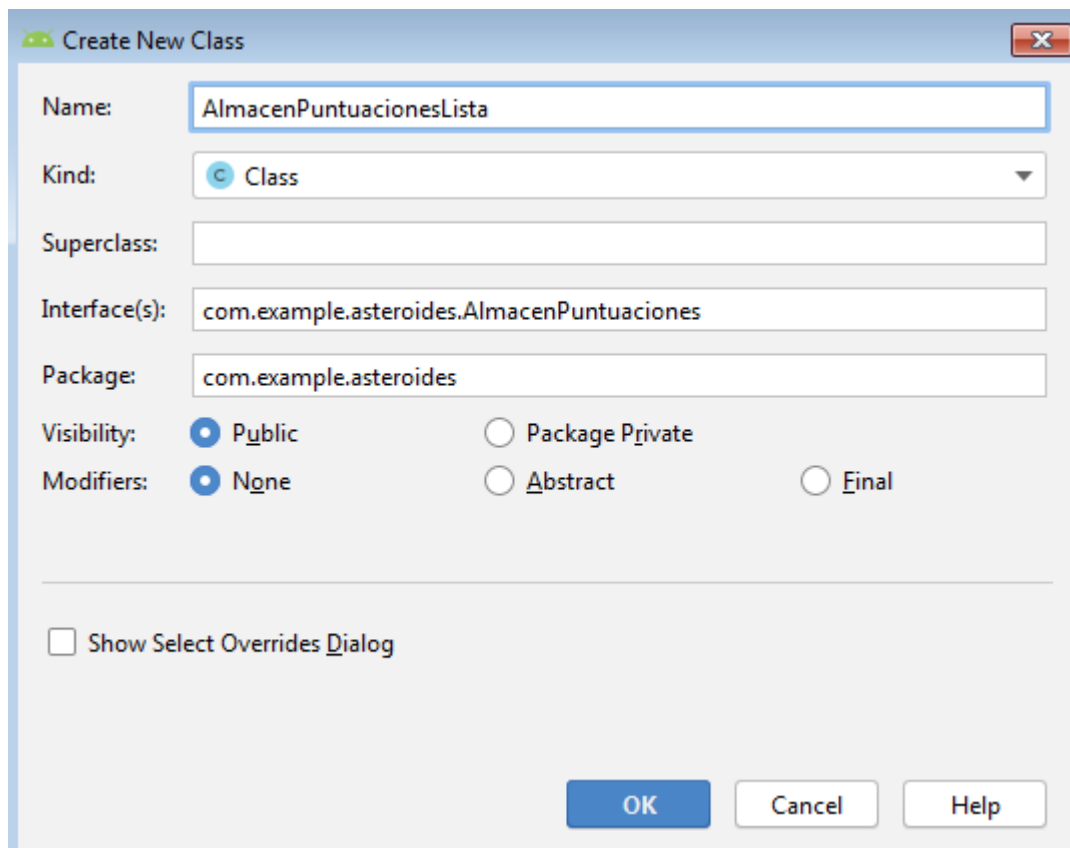
**Paso 4.** Introduce el código que se muestra a continuación:



**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: `AlmacenPuntuacionesLista`

Kind: `Class`

Superclass:

Interface(s): `com.example.asteroides.AlmacenPuntuaciones`

Package: `com.example.asteroides`

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:

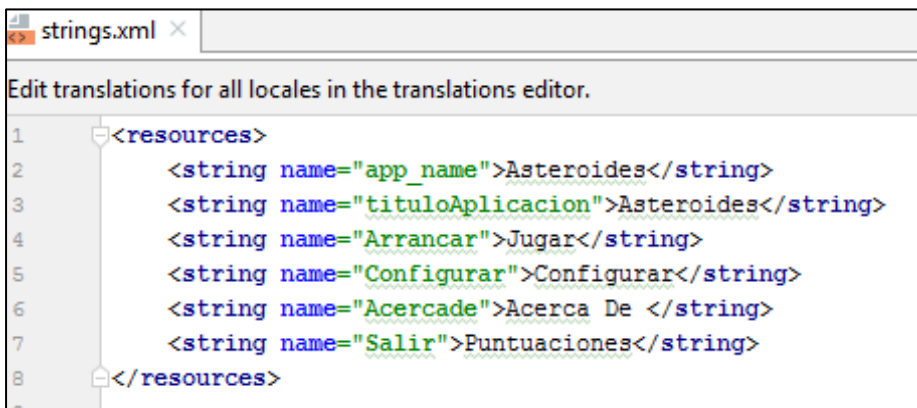
```
public class MainActivity extends AppCompatActivity {

    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`.



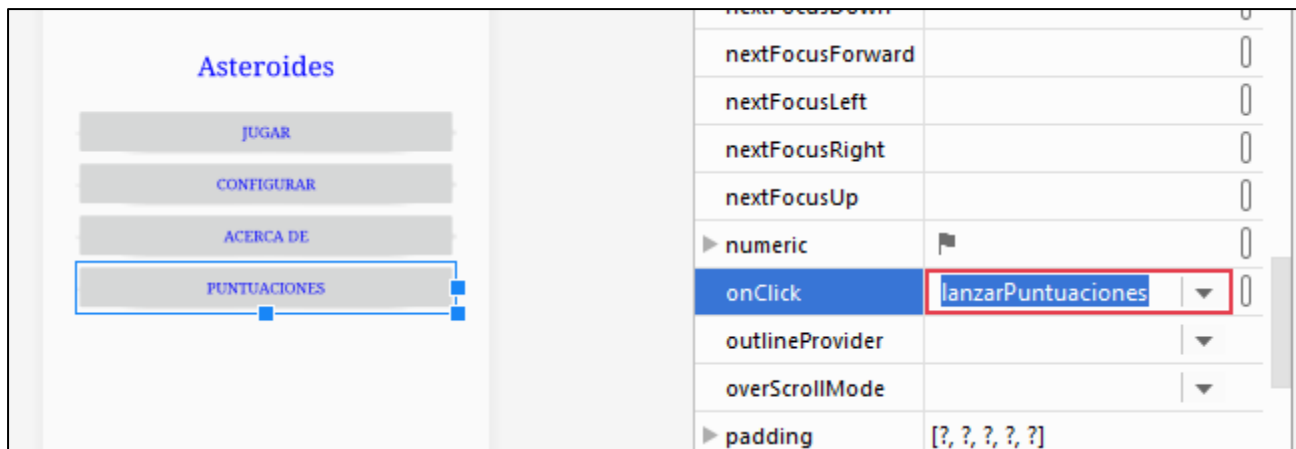
```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Asteroides</string>
    <string name="tituloAplicacion">Asteroides</string>
    <string name="Arrancar">Jugar</string>
    <string name="Configurar">Configurar</string>
    <string name="Acercade">Acerca De </string>
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