INDICE

1.1. Introducción

- ✓ Más información en https://developer.android.com/reference/android/widget/Spinner
- ✓ Ya vimos anteriormente como crear un Spinner, pero el enlace de los datos con Spinner se hacía a través de un atributo XML en la definición del Spinner en el Layout: **android:entries**.
- ✓ Ahora vamos a hacer lo mismo pero usando un adaptador.
- ✓ Cogeremos los datos desde un array en el código y desde recursos guardados en /res/values de tipo array.

1.2. Caso Base

- ✓ Lo que vamos a aprender en primer lugar es el funcionamiento básico del spinner.

- Como cargar datos en el spinner.
- Como modificar visualmente el aspecto de cada elemento del spinner.
- Como gestionar el evento de selección sobre un elemento del spinner.
- Como acceder al elemento seleccionado del spinner.
- Como borrar, modificar o añadir nuevos elementos al spinner.

✓ Posteriormente veremos otras posibilidades que nos brinda este control.

1.2.1.Preparando la Activity

✓ Vamos a crear una Activity que visualmente tenga este aspecto.

✓ Dentro del paquete **Adaptadores** vamos a crear una nueva 'Empty Activity' de nombre: **UD05_01_spinner_base** de tipo Launcher.

✓ Modificar el archivo **AndroidManifiest.xml** y añade un label a la activista.

1.2.2.Código del Layout

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
```

```
    android:layout_height="match_parent"
    tools:context=".UD04_01_spinner_base">

    <Button
        android:id="@+id/btnAlta"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_margin="5dp"
        android:text="Alta"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toStartOf="@+id/btnBaixa"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent" />

    <Button
        android:id="@+id/btnBaixa"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_margin="5dp"
        android:text="Baja"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toStartOf="@+id/btnModificar"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toEndOf="@+id/btnAlta" />

    <Button
        android:id="@+id/btnModificar"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_margin="5dp"
        android:text="Modificar"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toEndOf="@+id/btnBaixa" />

    <EditText
        android:id="@+id/edtEditElemLista"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="8dp"
        android:ems="10"
        android:hint="Texto Elemento Lista"
        android:inputType="textPersonName"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="@+id/guideline" />

    <androidx.constraintlayout.widget.Guideline
        android:id="@+id/guideline"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        app:layout_constraintGuide_percent="0.5" />

    <Spinner
        android:id="@+id/spnSpinner"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
```

```
        android:layout_marginTop="8dp"
        android:layout_marginEnd="16dp"
        android:popupBackground="@android:color/holo_orange_light"
        android:spinnerMode="dropdown"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

1.2.3.Cargando datos

- ✓ Lo primero que tenemos que lograr hacer es cargar los datos en la lista.
- ✓ Esto lo podemos hacer de dos formas:
  - Gráficamente, a través de la propiedad entreis del ListView.
  - Por código, haciendo uso de un ArrayAdapter.

1.2.3.1. Cargando datos desde el Layout

- ✓ Ya hemos visto como, empleando un recurso string-array guardado en un archivo de recurso en /res/values.

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string-array name="array_datos_spinner">
        <item>Mercurio</item>
        <item>Venus</item>
        <item>Terra</item>
        <item>Marte</item>
        <item>Xúpiter</item>
        <item>Saturno</item>
        <item>Urano</item>
        <item>Neptuno</item>
    </string-array>

</resources>
```

1.2.3.2. Cargando datos desde el código

- ✓ Nota: Quitamos la carga de datos si la tenemos en la propiedad 'entries' del Spinner.
- ✓ Partimos del ejercicio anterior en el que está definido un array de strings, en un archivo de /res/values:

Public constructors

ArrayAdapter(Context context, int resource)

Constructor

ArrayAdapter(Context context, int resource, int textViewResourceId)

Constructor

ArrayAdapter(Context context, int resource, T[] objects)

Constructor.

ArrayAdapter(Context context, int resource, int textViewResourceId, T[] objects)

Constructor.

ArrayAdapter(Context context, int resource, List<T> objects)

Constructor

ArrayAdapter(Context context, int resource, int textViewResourceId, List<T> objects)

Constructor

- ✓ Vamos a emplear el tercer constructor.
- ✓ Necesitamos indicarle al layout que va a emplear para visualizar los elementos del spinner. Podemos emplear cualquiera siempre que tenga al menos un TextView.
- ✓ Android proporciona ya unos predeterminados que podemos emplear, concretamente: **android.R.layout.simple_spinner_item** (podéis ver el diseño desde Android Studio pulsando la tecla Control y clicar con el ratón sobre simple_spinner_item).

  Por otra banda necesitamos algún tipo de array que guarde los datos.

  Nota: Indicar que el spinner tiene dos views diferentes:

  - El view que visualiza el elemento que se encuentra en el spinner.
  - El view que aparece cuando pulsamos en el spinner y se despliegan todos los elementos. Cada uno de ellos está en una view.

```
package com.example.adaptadores;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.Spinner;

public class UD04_01_spinner_base extends AppCompatActivity {

    private void cargarDatos(){

        String[]datos = getResources().getStringArray(R.array.array_datos_spinner);
        ArrayAdapter<String> adapter = new ArrayAdapter<>(this, an-
droid.R.layout.simple_spinner_item,datos);
```

```
        // adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        // View que aparece cuando pulsamos en el spinner

        Spinner spinner = findViewById(R.id.spnSpinner);
        spinner.setAdapter(adapter);
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_u_d04_01_spinner_base);

        cargarDatos();
    }
}
```

- ✓ Línea 5: Como vemos obtenemos los datos del string-array creado en /res/values en el punto anterior.
- ✓ Línea 6: Creamos un adaptador empleando un layout del S.O. y mandamos los datos.
- ✓ Línea 8: Podemos cambiar el layout que se va cargar al desplegar los elementos del spinner.
- ✓ Línea 11: Asociamos el adaptador al spinner (si no hacemos esto, no se cargará ningún dato.

- ✓ Podemos hacer una variante de este código y hacer que los datos vengan de un ArrrayList:

```
    private void cargarDatos(){

    ArrayList<String> datos = new ArrayList<>(Arrays.asList("Valor 1","Valor 2", "Valor 3"));
    ArrayAdapter<String> adapter = new ArrayAdapter<>(this, android.R.layout.simple_spinner_item,datos);

    // adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
    // View que aparece cuando pulsamos en el spinner

    Spinner spinner = findViewById(R.id.spnSpinner);
    spinner.setAdapter(adapter);
}
```

1.2.4.Personalizando el diseño

- ✓ Antes vimos que el segundo parámetro tenemos que enviarle el id del layout que va a representar cada uno de los elementos de la lista.

  Nosotros podemos crear nuestro propio Layout.

  Por ejemplo:

  Layout: elemento_spinnerview

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ElementoSpinner">
```

```
    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="VALOR:"
        android:textColor="@android:color/holo_purple"
        android:textStyle="bold"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/txvElemSpinner"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:text="TextView"
        app:layout_constraintStart_toEndOf="@+id/textView"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Ahora vamos a decirle al adaptador que cargue este diseño, pero como dentro del mismo hay dos TextView, tenemos que decirle cual de ellos debe emplear para pasar el dato de la fuente de datos al view.

Para eso empleamos otro de los construtores vistos anteriormente:

```
private void cargarDatos(){
        ArrayList<String> datos = new ArrayList<>(Arrays.asList("Valor 1","Valor 2", "Valor 3"));
        ArrayAdapter<String> adapter = new ArrayAdapter<>(this,
R.layout.elemento_spinnerview,R.id.txvElemSpinner,datos);

        Spinner spinner = findViewById(R.id.spnSpinner);
        spinner.setAdapter(adapter);
    }
```

Línea 3: Fijarse en dos cosas: Cargamos el layout diseñado por nosotros y enviamos como tercer parámetro el id del textview que se encuentra dentro del layout.

1.2.5.Gestionando el evento SELECT sobre un elemento del spinner

✓ A diferencia del ListView en este sí que podemos capturar el evento de selección.

Para gestionar el evento Click debemos emplear la interface OnItemSelectedListenerinterface.

NOTA IMPORTANTE: Cuando pulsamos en un elemento de la lista este no queda seleccionado por lo que no podemos emplear ningún método o interface que indique 'selected'.

```
private void xestionarEventos() {
    Spinner spinner = findViewById(R.id.spnSpinner);
    spinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
        @Override
        public void onItemSelected(AdapterView<?> adapterView, View view, int i, long l) {
            Toast.makeText(getApplicationContext(), adapter-
View.getItemAtPosition(i).toString(),Toast.LENGTH_SHORT).show();
            EditText et = findViewById(R.id.edtEditElemLista);
```

```
            et.setText(adapterView.getItemAtPosition(i).toString());
        }

        @Override
        public void onNothingSelected(AdapterView<?> adapterView) {

        }
    });

}
```

✓ Línea 5: En el método de la interface OnItemSelectedListener va a haber como parámetros:

- El AdapterView, a través del cual:
  - Podemos acceder a los elementos que están cargados en la lista.
  - Podemos recuperar la referencia al ArrayAdapter.
- El View que fue pulsado (recordar que si estamos haciendo el ejemplo, el view ahora mismo es el layout que hicimos con dos textview.
- Y es la posición del elemento seleccionado.
- Y es un identificador de fila (que no vamos a utilizar).

El anterior es equivalente y esto si tenemos acceso a los array de datos:

```
public class UD04_01_spinner_base extends AppCompatActivity {

    ArrayList<String> datos = new ArrayList<>(Arrays.asList("Valor 1","Valor 2", "Valor 3"));
    private void cargarDatos(){

        ArrayList<String> datos = new ArrayList<>(Arrays.asList("Valor 1","Valor 2", "Valor 3"));
        ArrayAdapter<String> adapter = new ArrayAdapter<>(this, an-
droid.R.layout.simple_spinner_item,datos);

        Spinner spinner = findViewById(R.id.spnSpinner);
        spinner.setAdapter(adapter);
    }

    private void xestionarEventos() {
        Spinner spinner = findViewById(R.id.spnSpinner);
        spinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> adapterView, View view, int i, long l) {
                Toast.makeText(getApplicationContext(), da-
tos.get(i).toString(),Toast.LENGTH_SHORT).show();
                EditText et = findViewById(R.id.edtEditElemLista);
                et.setText(adapterView.getItemAtPosition(i).toString());
            }

            @Override
            public void onNothingSelected(AdapterView<?> adapterView) {

            }
        });

    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_u_d04_01_spinner_base);
```

```
        cargarDatos();
    }
}
```

Línea 3: Definimos el array a nivel de Activity para poder acceder a él desde cualquier parte.

Línea 18: Accedemos a la posición delo array.

Curiosidad: El parámetro 'view' nos permite acceder a los views que están dentro del layout que diseñamos.

Por ejemplo:

```
private void xestionarEventos() {
    Spinner spinner = findViewById(R.id.spnSpinner);
    spinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
        @Override
        public void onItemSelected(AdapterView<?> adapterView, View view, int i, long l) {
            TextView tv = view.findViewById(R.id.txvElemSpinner);
            tv.setTextColor(Color.RED);

            Toast.makeText(getApplicationContext(), datos.get(i).toString(),Toast.LENGTH_SHORT).show();
        }

        @Override
        public void onNothingSelected(AdapterView<?> adapterView) {

        }
    });

}
```

1.2.6.Accediendo al elemento seleccionado de la lista

✓ Necesitamos acceder a los métodos getSeletedItem()ye getSelectedItemPosition que se encuentran en el AdapterView del Spinner.

Como curiosidad, indicar que también podemos acceder al View seleccionado llamando al método getSelectedView().

✓ En el ejemplo, vamos a hacer que cuando pulsemos sobre el EditText, aparezca la posición y el texto del elemento seleccionado en el Spinner:

```
findViewById(R.id.edtEditElemLista).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Spinner spinner = findViewById(R.id.spnSpinner);
                if (spinner.getSelectedItem()==null) return;    // Nada seleccionado
                Toast.makeText(getApplicationContext(), spinner.getSelectedItem().toString() + " EN LA
POSICION " + String.valueOf(spinner.getSelectedItemPosition()),Toast.LENGTH_SHORT).show();

                View layout_elem_spinner = spinner.getSelectedView();   // Podemos acceder al view selec-
cionado
                TextView tv = layout_elem_spinner.findViewById(R.id.txvElemSpinner);
                tv.setTextColor(Color.RED);
            }
```

```
        });
```

1.2.7.Operaciones sobre el spinner

- ✓ Podemos hacer alta, bajas y modificaciones.
  - Dichas operaciones las podemos hacer sobre el Array donde tenemos guardados los datos (siempre que este definido a nivel de Activity o de forma global) o sobre el adaptador asociado al Spinner.
- ✓ El ArrayAdapter dispone de métodos para añadir, borrar y modificar elementos de la lista.
- ✓ Por ejemplo:
  - add(T object): Añade un elemento a la lista de elementos.
  - addAll(Collection<? extends T> collection): Añade una colección.
  - getCount(): Devuelve el número de elementos de la lista.
  - getItem(int position): Devuelve el elemento de la lista en base a su posición (empieza en 0).
  - insert(T object, int index) : Añade un elemento en la posición indicada.
  - remove(T object): Elimina un elemento de la lista.
  - clear(): Borra todos los elementos de la lista.

Los veremos de dos formas:

1.2.7.1. Operación de baja

Sabemos, como vimos en el ejemplo anterior, como obtener la posición del elemento seleccionado.

1.2.7.1.1. Accediendo al adaptador

Suponemos que no tenemos acceso al array de datos (por estar definido localmente, por ejemplo).

```
findViewById(R.id.btnBaixa).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Spinner spinner = findViewById(R.id.spnSpinner);
                if (spinner.getSelectedItemPosition()==AdapterView.INVALID_POSITION) return;

                ArrayAdapter<String> adapter = (ArrayAdapter)spinner.getAdapter();
                String datoEliminar = (String)spinner.getSelectedItem();
                adapter.remove(datoEliminar);

            }
        });
```

Línea 7: Al no tener acceso al array de datos, cualquier operación la tenemos que hacer a través del adaptador. Por lo tanto necesitamos obtener una referencia al adaptador de la ListView.

Línea 8: A través del adaptador podemos acceder al elemento seleccionado por la posición que tenemos guardada previamente en el método onItemClick.

Línea 9: Eliminamos el elemento del adaptador. Automáticamente ya se refresca el ListView y desaparece el elemento.

1.2.7.1.2. Accediendo al array de datos

✓ En este caso podemos acceder al array de datos que está cargado en el adaptador (por tenerlo definido a nivel de Activity, no localmente en un método).

Revisar que en la carga de datos empleéis el array definido a nivel de Activity.

```
        findViewById(R.id.btnBaixa).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Spinner spinner = findViewById(R.id.spnSpinner);
                if (spinner.getSelectedItemPosition()==AdapterView.INVALID_POSITION) return;

                ArrayAdapter<String> adapter = (ArrayAdapter)spinner.getAdapter();
                datos.remove(spinner.getSelectedItemPosition());
                adapter.notifyDataSetChanged();

            }
        });
```

Línea 8: Eliminamos el elemento del array de datos.

Línea 9: Necesitamos informar al ListView que tiene que recargar la lista, llamando al método notifyDataSetChanged() del adaptador.

1.2.7.2. Operación de Alta

1.2.7.2.1. Accediendo al adaptador

Suponemos que no tenemos acceso al array de datos (por estar definido localmente, por ejemplo).

```
        findViewById(R.id.btnAlta).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                EditText et = findViewById(R.id.edtEditElemLista);
                if (et.getText().toString().length()==0) return;

                ArrayAdapter adaptador =
(ArrayAdapter)((Spinner)findViewById(R.id.spnSpinner)).getAdapter();
                adaptador.add(et.getText().toString());

            }
        });
```

Línea 8: Al no tener acceso al array de datos, cualquier operación la tenemos que hacer a través del adaptador. Por lo tanto necesitamos obtener una referencia al adaptador del Spinner.

Línea 9: A través del adaptador podemos añadir un nuevo texto.

1.2.7.2.2. Accediendo al array de datos

En este caso podemos acceder al array de datos que está cargado en el adaptador (por tenerlo definido a nivel de Activity, no localmente en un método).

```
        findViewById(R.id.btnAlta).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                EditText et = findViewById(R.id.edtEditElemLista);
                if (et.getText().toString().length()==0) return;

                ArrayAdapter adaptador =
(ArrayAdapter)((Spinner)findViewById(R.id.spnSpinner)).getAdapter();
                datos.add(et.getText().toString());
                adaptador.notifyDataSetChanged();

            }
        });
```

Línea 8: Añadimos el elemento al array de datos.

Línea 9: Necesitamos informar al Spinner que tiene que recargar la lista, llamando al método notifyDataSetChanged() del adaptador.

1.2.7.3. Operación de Modificar

1.2.7.3.1. Accediendo al adaptador

Suponemos que no tenemos acceso al array de datos (por estar definido localmente, por ejemplo).

Lo más fácil en este caso es borrar y dar de alta el nuevo elemento.

```
        findViewById(R.id.btnModificar).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                EditText et = findViewById(R.id.edtEditElemLista);
                if (et.getText().toString().length()==0) return;

                Spinner spinner = findViewById(R.id.spnSpinner);
                ArrayAdapter adaptador =
(ArrayAdapter)((Spinner)findViewById(R.id.spnSpinner)).getAdapter();
                adaptador.remove(spinner.getSelectedItem());
                adaptador.insert(et.getText().toString(),spinner.getSelectedItemPosition());

            }
        });
```

Línea 9: Borramos el elemento seleccionado.

Línea 10: Añadimos el nuevo elemento.

NOTA IMPORTANTE: Si estamos trabajando con objetos (es decir, lo que guardamos como elemento de la lista es un objeto de una clase definida por nosotros, en el cual sobrescribimos el método toString para indicar lo que queremos visualizar), podemos no borrar y crear un nuevo objeto, ya que cuando obtenemos el elemento seleccionado estamos obteniendo la dirección de memoria donde ese objeto está alojado.

Por lo tanto si modificamos el objeto seleccionado es como si modificásemos el elemento del array de datos y podemos emplear la segunda forma (se explica a continuación) informando al adaptador que hubo un cambio en los datos.

1.2.7.3.2. Accediendo al array de datos

- ✓ En este caso podemos acceder al array de datos que está cargado en el adaptador (por tenerlo definido a nivel de Activity, no localmente en un método).

```
        findViewById(R.id.btnModificar).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                EditText et = findViewById(R.id.edtEditElemLista);
                Spinner spinner = findViewById(R.id.spnSpinner);

                if ((et.getText().toString().length()==0) ||
(spinner.getSelectedItemPosition()==AdapterView.INVALID_POSITION)) return;

                ArrayAdapter adaptador =
(ArrayAdapter)((Spinner)findViewById(R.id.spnSpinner)).getAdapter();
                datos.set(spinner.getSelectedItemPosition(),et.getText().toString());
                adaptador.notifyDataSetChanged();

            }
        });
```

Línea 10,11: Borramos y añadimos el elemento al array de datos.

- ✓ Si estamos trabajando con objetos (es decir, lo que guardamos como elemento de la lista es un objeto de una clase definida por nosotros, en el cual sobrescribimos el método toString para indicar lo que queremos visualizar), podemos no borrar y crear un nuevo objeto, ya que cuando obtenemos el elemento seleccionado estamos obteniendo la dirección de memoria donde ese objeto está alojado.

  Por lo tanto si modificamos el objeto seleccionado es como si modificásemos el elemento del array de datos y podemos emplear la segunda forma (se explica a continuación) informando al adaptador que hubo un cambio en los datos.

  Quedaría el siguiente código suponiendo que tenemos añadido al spinner un ArrayList<Clase> siendo <Clase> una clase creada por nosotros y con el método toString implementado.

```
findViewById(R.id.btnModificar).setOnClickListener(new View.OnClickListener() {
@Override
public void onClick(View view) {
```

```
         EditText et = findViewById(R.id.edtEditElemLista);
        Spinner spinner = findViewById(R.id.spnSpinner);

         if ((et.getText().toString().length()==0) || (spin-
ner.getSelectedItemPosition()==AdapterView.INVALID_POSITION)) return;

                ArrayAdapter<Clase> adaptador = (ArrayAdap-
ter)((Spinner)findViewById(R.id.spnSpinner)).getAdapter();
                Clase clase = spinner.getSelectedItemPosition());
                clase.propiedade = 'Valor modificado'
                adaptador.notifyDataSetChanged();

            }
        });
```

1.2.7.4. Operación Seleccionar

✓ En el spinner podemos seleccionar un elemento de la lista por programación fácilmente llamando al método setSelection(int pos), indicando en el parámetro la posición del elemento a seleccionar, teniendo en cuenta que la posición del primer elemento empieza en 0.

1.3. Otros métodos

✓ Para crear un ArrayAdapter también podemos emplear este código:

```
Spinner lista = findViewById(R.id.id_spinner);

ArrayAdapter<CharSequence>adaptador =
ArrayAdapter.createFromResource(this,R.array.planetas_UD02_01_spinner,android.R.layout.simple_list_item_1
);
         adaptador.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
lista.setAdapter(adaptador);
```

✓ En la línea 3 creamos un ArrayAdapter haciendo uso del método createFromResource. Dicho método recibe como parámetros:

- El contexto.
- Los datos del ArrayAdapter, que en este caso vienen de un recurso ya creado previamente.
- El Layout que va a tener asociado cada view (elemento da lista, que dependiendo del layout serán tanto views como TextViews).
- En la línea 4, establecemos el layout que va a tener el spinner cuando pulsemos sobre él y aparezcan todos los elementos de la lista.
- En la línea 5 asociamos el ArrayAdapter al Spinner.

✓ Podemos hacer lo mismo de esta otra forma:

```
ArrayAdapter<String> adapter = new
ArrayAdapter<String>(this,android.R.layout.simple_spinner_item,getResources().getStringArray(R.array.plan
etas_UD02_01_spinner));
```

Para referenciar al array tendríamos que hacer uso de la clase getResource () y obtener el tipo correspondiente (array).

- ✓ **Importante:**
- ✓ Si el array que se le envía al adaptador como fuente de datos no es dinámico (por ejemplo, un Array de Strings) no podremos hacer hacer ninguna operación sobre los datos.
- ✓ Cualquier modificación que se haga sobre los elementos de la lista haciendo uso del adaptador, lleva consigo la 'obligación' de informar al view de los cambios para que los refleje.

  Esto se consigue llamando a uno de estos dos métodos:

  - Método setNotifyOnChange(boolean notifyOnChange), con valor true en el parámetro.
  - Método notifyDataSetChanged.

    El segundo normalmente se llama cuando hay algún cambio estructural en la lista (remove, add,...). En teoría lo llamará el primer método automáticamente pero algunas veces no funciona y es necesario llamarlo explícitamente. Se debe llamar a uno de los dos, no a los dos.

- ✓ Por ejemplo, el código anterior se podría cambiar por:

```
ArrayAdapter<String> adapter = new
ArrayAdapter<String>(this,android.R.layout.simple_spinner_item);
adapter.addAll(getResources().getStringArray(R.array.planetas_UD02_01_spinner));
adapter.setNotifyOnChange(true);
```

- ✓ La gestión de eventos en este control incluye la interface que controla cuando un elemento de la lista es pulsado. Es la interface OnItemSelectedListener y se registra con el método setOnItemSelectedListener(AdapterView.OnItemClickListener l) del Spinner.

Al hacerlo, deberán implementarse estos dos métodos:

```
@Override
public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
}

@Override
public void onNothingSelected(AdapterView<?> parent) {

}
```

El segundo método se llama cuando no se selecciona ningún elemento y desaparece la lista.

El primero cuando se selecciona un elemento de la lista.

Lleva como parámetros:

- Adaptador asociado a la lista.
- La view que fue seleccionada (recordar que cuando se crea un adaptador se establece un layout para cada componente de la lista).
- La posición del elemento seleccionado en la lista (empieza en 0).
- Un identificador para la fila que fue seleccionada.

1.4. Caso práctico 1

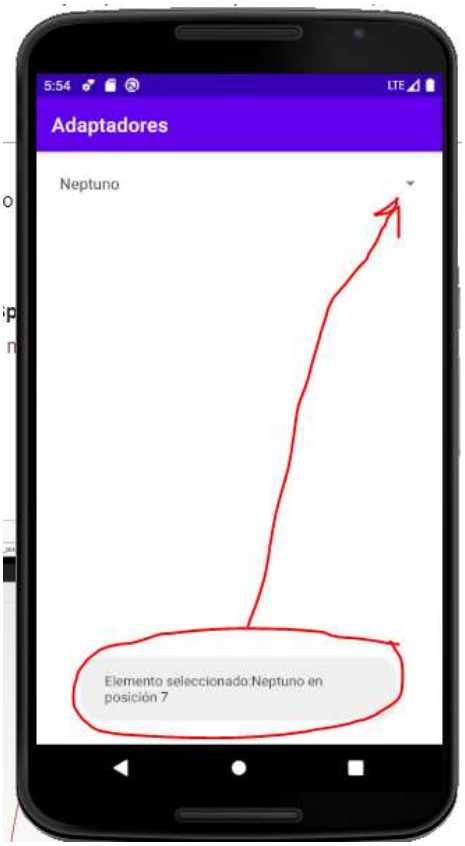
✓ Partimos que ya tenemos creado el proyecto inicial.

Se no lo tenemos creado antes, crearemos un nuevo paquete de nombre: **Adaptadores** como un subpaquete del paquete principal.

✓ Dentro del paquete **Adaptadores** crear una nueva 'Empty Activity' de nombre: **UD04_01_Spinners** de tipo Launcher.

Modificar el archivo **AndroidManifiest.xml** y añadir una label a la activity.

Spinner con Adapter



✓ Cada ítem del Spinner es tratado como una **View**, en este caso de tipo **TextView**.

1.4.1.El XML del layout

✓ Observar como la vista Spinner no tiene la entrada **android:entries**.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".UD04_01_Spinners">

    <Spinner
        android:id="@+id/spnLista_UD02_02_spinner"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        android:dropDownWidth="match_parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

✓ Partimos del ejercicio de spinner hecho anteriormente, en el que está definido un array de datos, en un archivo de /res/values:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name = "planetas_spinner">
        <item> Mercurio </item>
        <item> Venus </item>
        <item> Tierra </item>
        <item> Marte </item>
        <item> Júpiter </item>
        <item> Saturno </item>
        <item> Urano </item>
        <item> Neptuno </item>
    </string-array>

</resources>
```

✓ El código de la Activity:

```
package com.example.adaptadores;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;
```

```
public class UD04_01_Spinners extends AppCompatActivity {

    private void cargarLista() {

        Spinner lista = findViewById(R.id.spnLista_UD02_02_spinner);

        ArrayAdapter<String> adapter = new ArrayAdap-
ter<String>(this,android.R.layout.simple_list_item_1);
        adapter.addAll(getResources().getStringArray(R.array.planetas_spinner));
        lista.setAdapter(adapter);

        lista.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
                //String datoLista = (String)parent.getItemAtPosition(position);    // Podríamos obtener
el dato de esta forma (fijarse en el Cast)
                String datoLista = ((TextView) view).getText().toString();          // o de esta otra
forma
                Toast.makeText(getApplicationContext(),"Elemento seleccionado:" + datoLista + " en posi-
ción "+ String.valueOf(position),Toast.LENGTH_LONG).show();
            }

            @Override
            public void onNothingSelected(AdapterView<?> parent) {

            }
        });

    }
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_u_d04_01__spinners);

        cargarLista();

    }
}
```

- ✓ Líneas 20-22: Usamos una de las formas vistas para crear el adaptador y asociarlo al spinner.
- ✓ Línea 24: Registramos la interface que gestiona el evento de selección de un elemento de la lista.
- ✓ Línea 26-30: Hacemos del método onItemSelected() y sus parámetros para obtener el dato seleccionado de la lista.

1.4.2.Caso práctico 2: Usando un array estático en Java

- ✓ Dentro del paquete **Adaptadores** crear una nueva 'Empty Activity' de nombre: **UD04_02_Spinners** de tipo Launcher.

  Modificar el archivo **AndroidManifiest.xml** y añadir una etiqueta a la activity.

- ✓ En este caso cargaremos la fuente de datos de un array local.

✓ El código XML de la activity es:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".UD04_02_Spinners">

    <Spinner
        android:id="@+id/spnLista_UD04_02_Spinner"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

✓ Al final del código se explica cómo se enlaza la fuente de datos con el adaptador y este con el spinner.

✓ Observar las líneas marcadas.

```
package com.example.adaptadores;

import androidx.appcompat.app.AppCompatActivity;
```

```
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;

public class UD04_02_Spinners extends AppCompatActivity {

    private void cargarLista(){

        Spinner spinFrutas = (Spinner) findViewById(R.id.spnLista_UD04_02_Spinner);

        // Fuente de datos
        String[] frutas = new String[] { "Pera", "Manzana", "Plátano" };

        // Enlace del adaptador con los datos
        ArrayAdapter<String> adaptador = new ArrayAdapter<String>(this, an-
droid.R.layout.simple_spinner_item, frutas);

        // Opcional: Layout usado para representar los datos en el Spinner
        adaptador.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);

        // Enlace del adaptador con Spinner del Layout.
        spinFrutas.setAdapter(adaptador);

        // Escuchador
        spinFrutas.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent, View view, int pos, long id) {
                //Toast.makeText(getBaseContext(), "Seleccionaste: " + parent.getItemAtPosition(pos),
Toast.LENGTH_LONG).show();
                Toast.makeText(getBaseContext(), "Seleccionaste: " + ((TextView) view).getText(), To-
ast.LENGTH_LONG).show();

            }

            @Override
            public void onNothingSelected(AdapterView<?> arg0) {
                // TODO Auto-generated method stub
            }
        }); // Fin da clase anónima

    }


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_u_d04_02__spinners);

        cargarLista();
    }
}
```

- ✓ **Línea 23**: Definimos la fuente de datos de datos, en este caso un array estático.
- ✓ **Línea 26**: Definimos el adaptador de tipo ArrayAdapter. Le pasamos como parámetros:
  - Contexto,

- El int, identificador de recurso de layout que se va a usar para representar la Vista de Selección, en este caso usamos uno predefinido. El usuario puede experimentar con otros tipos (android.R.layout.... CTRL+ESPAZO) y ver que otras formas hay de presentar los datos de un Spinner.
- Un array de objetos, en este caso de Strings.
- Para ver otros constructores y métodos: http://developer.android.com/reference/android/widget/ArrayAdapter.html

✓ **Línea 29**: *setDropDownViewResource(int)*, indica cómo se va a representar cada uno de los ítems del Spinner. Usamos un layout ya predefinido. El usuario puede experimentar usando otros distintos, predefinidos o propios.

✓ **Línea 32**: establece el adaptador que subministra los datos al Spinner.

✓ **Línea 35**: El escuchador es el que cuando no se usaba un adaptador.

✓ **Líneas 38,39**: Las dos líneas hacen lo mismo, pero en el segundo caso observar como recogemos la vista (view) qué nos devuelve el evento al pulsar en un ítem del Spinner. Esa view es del tipo TextView y por eso hacemos un cast y luego ya le podemos aplicar métodos da clase TextView, como getText().

1.4.3.Caso práctico 3: Uso de un array dinámico

✓ Dentro del paquete **Adaptadores** crear una nueva 'Empty Activity' de nombre: **UD04_02_Spinners** de tipo Launcher. Modificar el archivo **AndroidManifest.xml** y añadir una etiqueta a la activity.

✓ En este caso cargaremos la fuente de datos de un array dinámico.

✓ El código XML de la activity es:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".UD04_03_Spinners">
    <Spinner
        android:id="@+id/spnLista_UD04_03_Spinner"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

- ✓ En los dos casos anteriores el contenido del Spinner es estático y se define en tiempo de compilación, no de ejecución.
- ✓ Si usamos arrays dinámicos podemos crear la fuente de datos en tiempo de ejecución antes de pasarla al adaptador.
- ✓ Así vamos a poder usar datos de ficheros, bases de datos, etc y crear una fuente de datos para un Spinner en tiempo de ejecución.

```
package com.example.adaptadores;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Spinner;
import android.widget.TextView;
import android.widget.Toast;

import java.util.ArrayList;

public class UD04_03_Spinners extends AppCompatActivity {

    private void cargarLista(){

        Spinner spinFroitas = (Spinner) findViewById(R.id.spnLista_UD04_03_Spinner);

        ArrayList<String> animais = new ArrayList<String>();

        animais.add("CABALLOS");
        animais.add("GATITOS");
        animais.add("PERROS");
        animais.add("RATONES");

        // Enlace del adaptador con los datos
        ArrayAdapter<String> adaptador = new ArrayAdapter<String>(this, an-
droid.R.layout.simple_spinner_item, animais);

        // Enlace del adaptador con el Spinner del Layout.
        spinFroitas.setAdapter(adaptador);

        // Escuchador
        spinFroitas.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
```

```
            @Override
            public void onItemSelected(AdapterView<?> parent, View view, int pos, long id) {
                //Toast.makeText(getBaseContext(), "Seleccionaches: " + parent.getItemAtPosition(pos),
Toast.LENGTH_LONG).show();
                Toast.makeText(getBaseContext(), "Seleccionaste: " + ((TextView) view).getText(), To-
ast.LENGTH_LONG).show();

            }

            @Override
            public void onNothingSelected(AdapterView<?> arg0) {
                // TODO Auto-generated method stub
            }
        }); // Fin de la clase anónima

    }


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_u_d04_03__spinners);

        cargarLista();
    }
```

- ✓ **Líneas 23-28**: En este caso los elementos del array se añaden en tiempo de ejecución al array **animales**, que este caso es del tipo **ArrayList**.
- ✓ O resto es exactamente igual.

1.4.4.Caso práctico 4: Creando un adaptador propio

- ✓ Dentro del paquete **Adaptadores** crear una nueva 'Empty Activity' de nombre: **UD04_04_Spinners** de tipo Launcher.

  Modificar el archivo **AndroidManifiest.xml** y añadir una etiqueta a la activity.

- ✓ Como indicamos desde el inicio, cuando creamos el Spinner también creamos un adaptador sobre el cual indicamos el layout de cada componente de la lista.
- ✓ Podemos crear los mismos el diseño que queramos con los datos que queramos que aparezcan en la lista.
- ✓ Para hacer lo anterior, tenemos que crear nuestro propio adaptador.
- ✓ En esta práctica creamos un layout formado por una imagen y un texto para cada elemento de la lista do spinner.

✓ Como o que queremos hacer un diseño específico para cada 'fila' de elementos de la lista / spinner, vamos a crear el Layout que va a representar cada fila:

Nombre Archivo del layout: activity_u_d04_04__spinners.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".UD04_04_Spinners">

    <ImageView
        android:id="@+id/imgVImaxen_UD04_04_Spinners"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
         />

    <TextView
        android:id="@+id/txtTexto_UD04_04_Spinners"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"

         />

    <Spinner
        android:id="@+id/spnLista_UD04_04_Spinner"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginTop="8dp"
```

```
        android:layout_marginEnd="8dp"
        android:layout_marginBottom="8dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</LinearLayout>
```

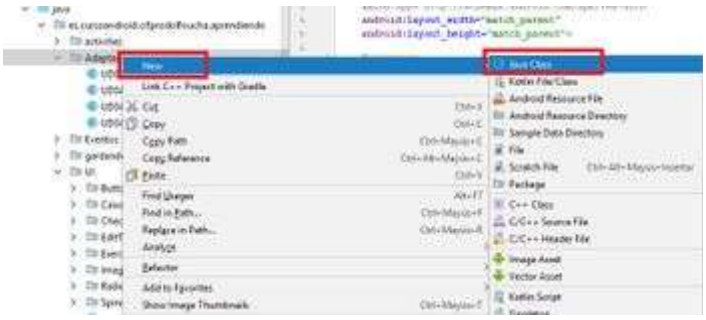
Como vemos, estamos definiendo un layout con una Imageview y un TextView. Pero podríamos poner cualquier layout con cualquier número de elementos y diseño.

Partimos de un ejercicio anterior donde ya tenemos definidos en un array los datos del ejercicio:

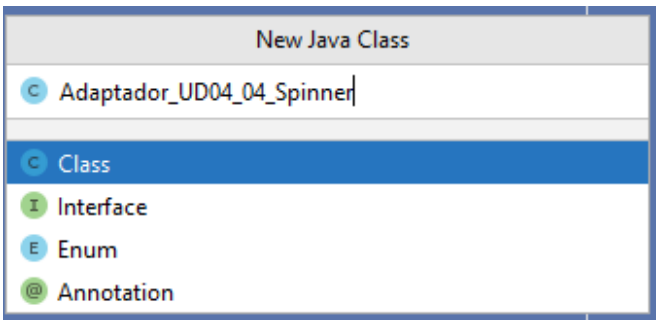
```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string-array name="planetas_UD02_01_spinner">
        <item>Mercurio</item>
        <item>Venus</item>
        <item>Terra</item>
        <item>Marte</item>
        <item>Júpiter</item>
        <item>Saturno</item>
        <item>Urano</item>
        <item>Neptuno</item>
    </string-array>

</resources>
```

✓ Ahora crearemos el Adaptador.

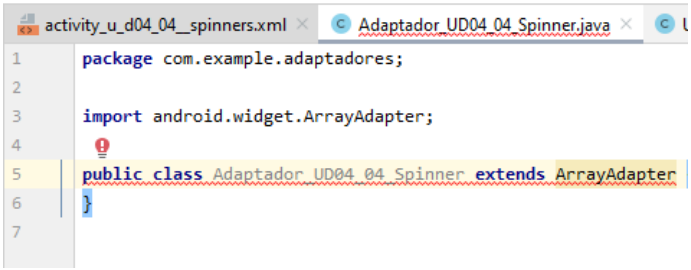
Creando un adaptador propio



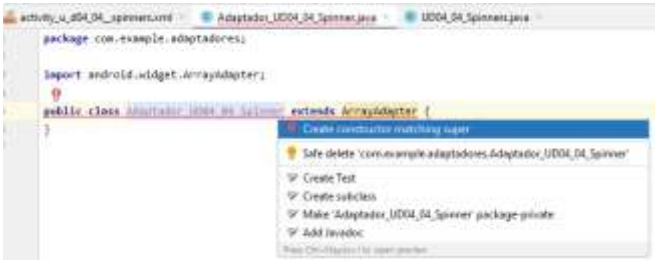
Creamos una nueva Java Class.



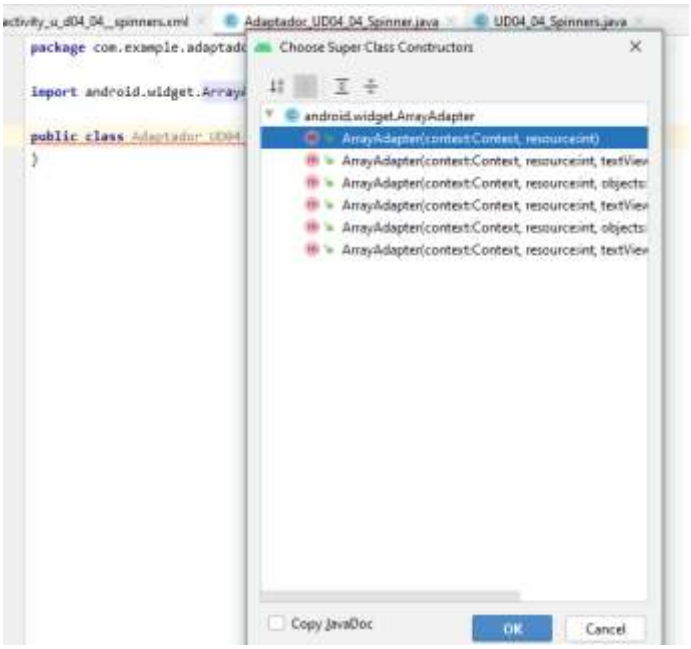
Le damos un nombre a la clase y hacemos que herede de la clase ArrayAdapter. En nuestro ejemplo se llama **Adaptador_UD04_04_Spinners**.



Aparecerá un error.



Debemos situarnos sobre la línea con error y teclear Alt+Enter. Escogemos la opción de 'Create constructor...'.



Dentro de los constructores posibles escogemos el primero. Esto da igual ya que el constructor lo vamos a personalizar para nuestras necesidades.

Después de hacer lo anterior tendremos un código como este:

Archivo: Adaptador_UD04_04_Spinners

```
public class Adaptador_UD04_04_Spinner extends ArrayAdapter {
    public Adaptador_UD04_04_Spinner(@NonNull Context context, int resource) {
        super(context, resource);
    }
}
```

✓ Ahora debemos tener claro que lo que vamos a 'enviar' a este adaptador van ser los datos necesarios para que 'construya' la lista de elementos.

En nuestro caso, queremos que haga una lista de elementos en el que cada elemento va a ser un texto (TextView) y una imagen (ImageView).

Por lo tanto tendremos que enviar un array de cadenas (para el texto) y algo para hacer una imagen. En el ejemplo vamos a enviarle imágenes guardadas en /res/drawable/ y por lo tanto serán un array de enteros (recordar que en /res/ todo se referencia por números haciendo uso de la clase R). Podríamos enviar URL y que la ImageView cargase la imagen de dicha URL.

Modificamos el constructor para que recoja estos datos. Como tendremos que hacer uso de ellos en otros métodos, crearemos propiedades locales en la clase:

```
public class Adaptador_UD04_04_Spinner extends ArrayAdapter {
    int[] imagenes;   // Lista de imagenes a cargar. Recordar que en res todo son números  Se fueran url
podríamos cambiarlo por String[]
    String[] textos;
    Context mContext;
    public Adaptador_UD04_04_Spinner(@NonNull Context context, int resource) {
        super(context, R.layout.activity_u_d04_04__spinners);

        this.imagenes = imagenes;
        this.textos = textos;
        this.mContext = context;

    }
}
```

Fijarse en la línea 8, que enviamos al constructor de la clase ArrayAdapter al layout que va a 'guardar' cada elemento de la lista en el spinner.

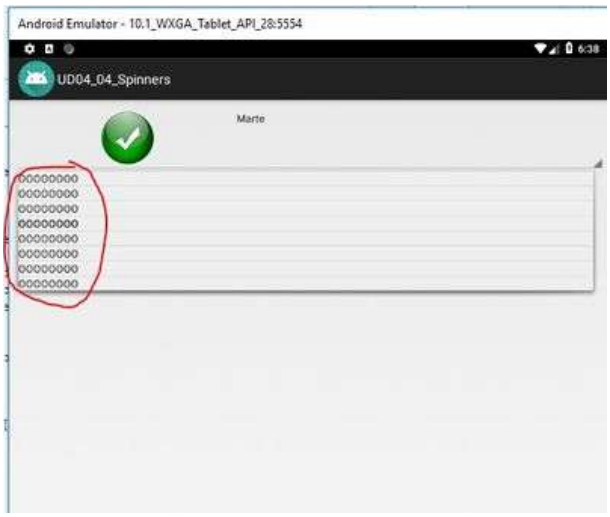
- ✓ Ahora, para que funcione el adaptador tenemos que sobrescribir al menos tres métodos:
  - int **getCount**(): Devuelve el número de elementos de la lista. Fijarse que en nuestro caso, el número de elementos de la lista lo sabemos por cualquiera de los dos arrays donde guardamos los datos. Si creamos cuatro elementos, la propiedad textos.length será cuatro.
  - View **getView**(int position, @Nullable View convertView, @NonNull ViewGroup parent): Este método tiene que devolver el view de cada elemento de la fila. Fijarse que este 'view' en nuestro caso será un LinearLayout formado por dos Views (un TextView y un ImageView). Es el view que visualiza el elemento escogido de la lista.
  - View **getDropDownView**(int position, @Nullable View convertView, @NonNull ViewGroup parent): Devuelve el view que conforma cada elemento de la lista cuando pulsamos sobre la misma (la lista desplegable). En el ejemplo vamos a mostrar el mismo que en getView (pero no tendría por qué ser así).

Fijarse que en el caso de los métodos que devuelve un view, podemos hacer uno dinámicamente y devolverlo.

Por ejemplo, si hago esto:

```
@Override
public View getDropDownView(int position, @Nullable View convertView, @NonNull ViewGroup parent) {
    TextView prueba = new TextView(mContext);
    prueba.setText("00000000");
    return prueba;
}
```

Aparece esto:



Esto sería útil si queremos que cuando escojamos un elemento aparezca una información y cuando este escogido aparezca otra. Más información en este enlace.

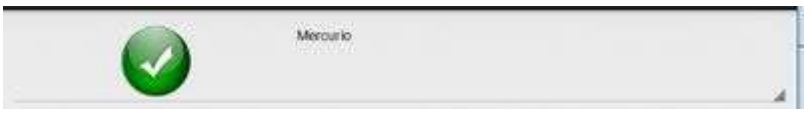
En nuestro caso queremos que visualmente aparezca el mismo cuando tenemos un elemento escogido de la lista, como cuando aparece el desplegable de la misma.

Como el view que representa el elemento seleccionado de la lista lo obtenemos del método getView (como comentamos antes) el código quedaría así:

```
@Override
public int getCount() {
    return textos.length;       // Podríamos poner también el número de imágenes.
}

@Override
public View getDropDownView(int position, @Nullable View convertView, @NonNull ViewGroup parent) {
    return getView(position,convertView,parent);
}

@NonNull
@Override
public View getView(int position, @Nullable View convertView, @NonNull ViewGroup parent) {
    return super.getView(position, convertView, parent);
}
```

- Línea 3: Devolvemos el número de textos enviados como parámetro del constructor.
- Línea 8: Devolvemos el view del método **getView** que vamos a modificar.

✓ Ahora viene los más importante, que es crear el view que conforma el elemento seleccionado en el Spinner:



El proceso para hacerlo es el siguiente:

Nos fijamos en la definición del método getView: public View getView(int position, @Nullable View **convertView**, @NonNull ViewGroup parent) {

Ese objeto de la clase View va a 'llevar' el layout que tenemos definido para cada elemento de la fila.

Y los datos con los que vamos a llenar dicho layout van a ir en el método setTag ya comentado.

Ese objeto que aparece como parámetro (convertView) representa el conjunto de elementos gráficos que conforman cada fila del spinner.

La primera vez que se llama, ese objeto vale 'null' y debemos de hacer que sea igual al layout en el que definimos la imagen y el texto que conforman cada elemento de la lista.

La segunda y siguientes veces, ese dato ya vendrá con el layout asociado.

✓ Lo **primero** es cargar el layout en el 'convertView'.

Para eso tenemos que pasar el layout de un archivo XML a un objeto que pueda ser empleado en programación.

Eso se hace haciendo un 'inflate':

```
LayoutInflater mInflater = (LayoutInflater) mContext.
        getSystemService(Context.LAYOUT_INFLATER_SERVICE);
convertView = mInflater.inflate(R.layout.activity_u_d04_04__spinners, parent, false);
```

Ya analizaremos en detalle este código. Por ahora quedar con la idea de que estas líneas hacen que el LinearLayout con botón y la imagen pasa a ser un objeto que se guarda en convertView.

Como dijimos antes, esto solamente hay que hacerlo si el convertView viene con un valor null.

```
@Override
public View getView(int position, @Nullable View convertView, @NonNull ViewGroup parent) {

    if (convertView == null) {

        LayoutInflater mInflater = (LayoutInflater) mContext.
                getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        convertView = mInflater.inflate(R.layout.activity_u_d04_04__spinners, parent, false);
    }
```

✓ **Segundo**, debemos pasar los datos a cada uno de los elementos gráficos que se encuentran en el objeto 'convertView'.

Para eso podemos emplear el método findViewById en el convertView. Lo único que tenemos que hacer es buscar cada componente gráfico y asociarle el dato que sabemos cuál es por el parámetro 'int position', que indica la posición del elemento de la lista seleccionado.

```
@Override
public View getView(int position, @Nullable View convertView, @NonNull ViewGroup parent) {
    //return super.getView(position, convertView, parent);

    if (convertView == null) {

        LayoutInflater mInflater = (LayoutInflater) mContext.
                getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        convertView = mInflater.inflate(R.layout.activity_u_d04_04__spinners, parent, false);
    }
    ((Image-
View)convertView.findViewById(R.id.imgVImaxen_UD04_04_Spinners)).setImageResource(imagenes[position]);
    ((TextView) convertView.findViewById(R.id.txtTexto_UD04_04_Spinners)).setText(textos[position]);

    return convertView;


}
```

Nota: Recordar que podemos emplear los métodos getTag() y setTag() para enviar datos u asociarlos a un view.

✓ Una vez que tenemos el adaptador creado tendremos que ir a la activity es asociar el spinner a dicho adaptador.

Como no tenemos muchas imágenes en /res/ vamos a asociar la misma imagen a todos los elementos de la lista. Partimos que tenemos una imagen en res de nombre: R.drawable.ok (se no es el caso poned otra cualquiera).

Archivo: UD04_04_Spinners

```
package com.example.adaptadores;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.widget.Spinner;

public class UD04_04_Spinners extends AppCompatActivity {

    private void cargarLista(){
        String datos[] = getResources().getStringArray(R.array.planetas_spinner);
        int[] imagenes = new int[datos.length];
        for(int cont=0;cont<datos.length;cont++){
            imagenes[cont]=R.drawable.icon_ok;
        }

       // Spinner lista = findViewById(R.id.txtTexto_UD04_04_Spinners);

        Spinner lista = findViewById(R.id.spnLista_UD04_04_Spinner);

        Adaptador_UD04_04_Spinner miAdaptador = new Adaptador_UD04_04_Spinner(this,imagenes,datos);
        lista.setAdapter(miAdaptador);

    }
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_u_d04_04__spinners);

        cargarLista();
    }
}
```