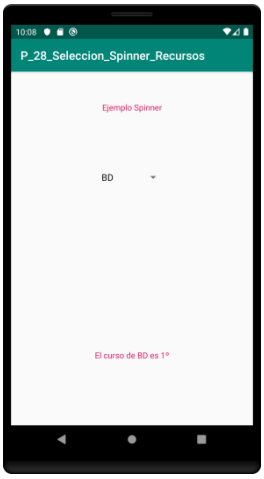
Ejercicio 1

Habrás observado que en la aplicación P_28_Seleccion_spinner_recursos (y en todas en las que hemos utilizado la vista spinner), nada más abrirse aparece ya seleccionada la primera opción del Spinner y por tanto se muestra el resultado de esa opción:

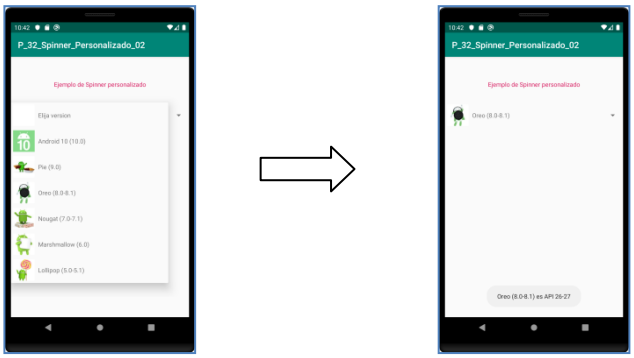


Corrige dicho comportamiento.

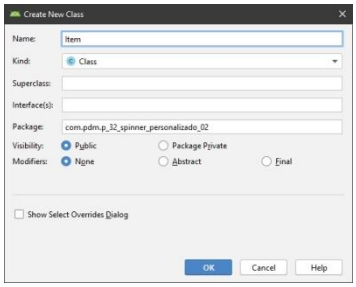
Pista: Añade una primera opción que sea "Elija materia" y un primer curso que sea "Elija materia".

Ejercicio 4: Proyecto P_32_Spinner_Personalizado_02

1. Crea un nuevo proyecto con Spinner personalizado mediante "*adapter*" propio.



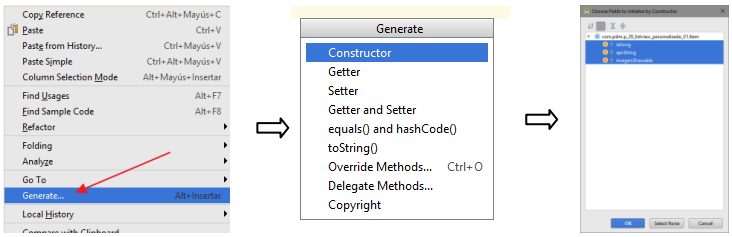
2. Los pasos son:
   a. Diseñar el layout propio del item de la lista usando ConstraintLayout.
   b. Si el item de la lista contiene objetos diversos (imágenes, textos,…), para trabajar mejor, nos **crearemos una clase** que lo defina. El camino más corto, con el paquete seleccionado, botón derecho, New, Java Class:



Completamos código:

```
public class Item {
   /*
    * simplemente es una clase que contendrá las variables necesarias para
    * almacenar los elementos que más tarde queremos que aparezcan en el
    * Spinner. En nuestro caso:
    * long id: posición del ítem dentro de la lista
    * String api: texto que aparece en el ítem
    * Drawable imagen: imagen que aparece en el ítem
    */
   long id;
   String api;
   Drawable imagen;
}
```

Para el resto de código haremos uso de los asistentes de generación de código de AS, "click" derecho en el editor de código:

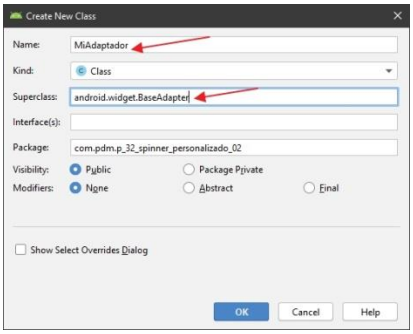


Haremos lo mismo para generar los "getter's". El código queda:

```
public class Item {
   /*
    * simplemente es una clase que contendrá las variables necesarias para
    * almacenar los elementos que más tarde queremos que aparezcan en el
    * ListView. En nuestro caso:
    * long id: posición del ítem dentro de la lista
    * String api: texto que aparece en el ítem
    * Drawable imagen: imagen que aparece en el ítem
    */
   long id;
   String api;
   Drawable imagen;

   public Item(long id, String api, Drawable imagen) {
      this.id = id;
      this.api = api;
      this.imagen = imagen;
   }
   public long getId() {
      return id;
   }
   public String getApi() {
      return api;
   }
   public Drawable getImagen() {
      return imagen;
   }
}
```

c. Personalizar el Adapter usando el layout propio y la clase anterior

Creamos una clase para nuestro adapter:



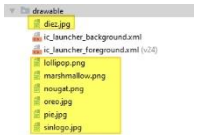
Admitimos la sugerencia de implementar los métodos que faltan y completamos con el código señalado en rojo (intenta entenderlo desde los comentarios):

```
public class MiAdaptador extends BaseAdapter {
  /*
  * Esta clase contiene dos atributos: de tipo Activity el primero y de tipo
  * ArrayList<ItemLista> el segundo. Ambos son pasados al constructor para
  * inicializar el adapter. El atributo Activity es necesario para poder
  * generar el layout que hemos creado anteriormente para nuestros item en el
  * Spinner. El atributo ArrayList de items contiene los elementos que se
  * mostrarán.
  */

  private final Activity contexto;
  private final ArrayList<Item> lista;

  public MiAdaptador(Activity contexto, ArrayList<Item> lista) {
    this.contexto = contexto;
    this.lista = lista;
  }

  @Override
  public int getCount() {
    return lista.size();
  }

  @Override
  public Object getItem(int position) {
    return lista.get(position);
  }

  @Override
  public long getItemId(int position) {
    return lista.get(position).getId();
  }

  @Override
  public View getView(int position, View convertView, ViewGroup parent) {
    /*
    * Debe “inflarse” el layout XML que hemos creado.
    * Esto consiste en  consultar el XML de nuestro layout y crear
    * e inicializar la estructura de objetos java equivalente. Para ello,
    * crearemos un nuevo  objeto LayoutInflater y
    * generaremos la estructura de objetos mediante su método inflate(id_layout).
```

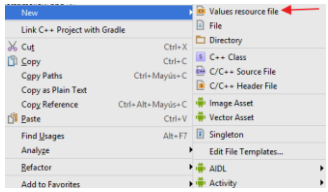
```
        */
        LayoutInflater inflater = contexto.getLayoutInflater();
        convertView = inflater.inflate(R.layout.layout_item, null, true);

        /*
        * Una vez que la vista para el item está preparada recuperamos el ítem que
        * vamos a mostrar utilizando el segundo parámetro que recibe el método getView
        * y el ArrayList que tenemos con los items.  A continuación vamos
        * recuperando los componentes de la vista y rellenándolos con los datos adecuados.
        */
        Item item = lista.get(position);
        TextView textView = convertView.findViewById(R.id.textView2);
        textView.setText(item.getApi());
        ImageView imageView = convertView.findViewById(R.id.imageView);
        imageView.setImageDrawable(item.getImagen());

        // Finalmente devolvemos la vista terminada de configurar.
        return convertView;
    }
}
```

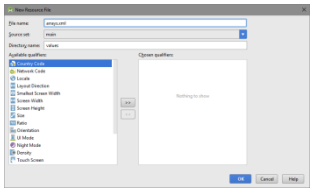
Nota: AS nos advierte con un "warning" que deberíamos usar el patrón ViewHolder. Ignóralo de momento, más adelante veremos que es.

d. En este ejercicio, nos interesa un [array de drawables] al que llamaremos logos:



Creamos un nuevo recurso tipo "values" llamado arrays.xml (o añadimos si ya tenemos el fichero):





```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <array name="logos">
        <item>@drawable/sinlogo</item>
        <item>@drawable/diez</item>
        <item>@drawable/pie</item>
        <item>@drawable/oreo</item>
        <item>@drawable/nougat</item>
        <item>@drawable/marshmallow</item>
        <item>@drawable/lollipop</item>
    </array>
</resources>
```

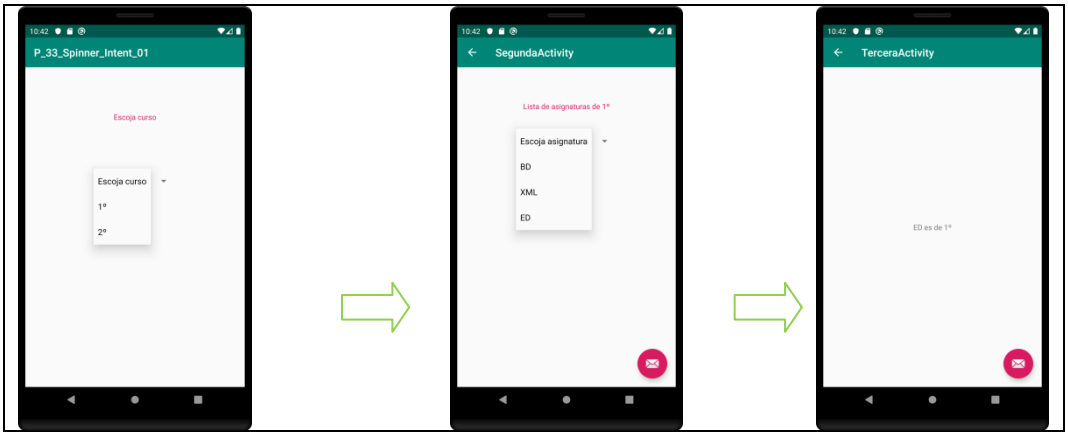
e. En MainActivity que contiene el Spinner asignamos el Adapter personalizado:

```
public class MainActivity extends AppCompatActivity {
    int posicionSeleccionada;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        // Accedemos a los arrays de string y drawable
        final Resources resources = getResources();
        final String[] version = resources.getStringArray(R.array.version);
        final String[] num_api = resources.getStringArray(R.array.num_api);
        TypedArray logos = resources.obtainTypedArray(R.array.logos);
        /*
         * Construimos la lista de nuestro adapter asociando cada item
         * con su número de api y su imagen
         */
        ArrayList<Item> lista = new ArrayList<>();
        for (int i = 0; i < 6; i++) {
            lista.add(new Item(i + 1, version[i], logos.getDrawable(i)));
        }
        logos.recycle();
        MiAdaptador miAdaptador = new MiAdaptador(this, lista);
        Spinner spinner = findViewById(R.id.spinner);
        spinner.setAdapter(miAdaptador);
        spinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
            @Override
            public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
                if (position!=0) {
                    String resu = version[position] + " es " + num_api[position];
                    Toast.makeText(getApplicationContext(), resu, Toast.LENGTH_SHORT).show();
                }
            }
            @Override
            public void onNothingSelected(AdapterView<?> parent) {

            }
        });
    }
}
```

3. Recuerda drawables por tamaño usando plugin instalado.

Ejercicio 2: Proyecto P_33_Spinner_Intent_01

1. La aplicación tiene tres actividades:
   a. MainActivity: basada en plantilla EmptyActivity
   b. SegundaActivity: basada en plantilla BasicActivity y con "padre jerárquico" MainActivity
   c. TerceraActivity: basada en plantilla BasicActivity y con "padre jerárquico" MainActivity



2. En la tercera actividad, observa el comportamiento distinto de  y 

Ejercicio 2: P_34_Spinner_Intent _02

1. La aplicación tiene dos actividades, basadas ambas en la plantilla Empty. Desde MainActivity se lanza SegundaActivity y desde esta se vuelve a MainActivity (recuerda startActivityForResult)

MainActivity

SegundaActivity