UNIVERSIDAD DON BOSCO
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA EN COMPUTACIÓN
DESARROLLO DE SOFTWARE PARA MÓVILES DSM104 G01T

COMPONENTES EN ANDROID

Tabla de contenido

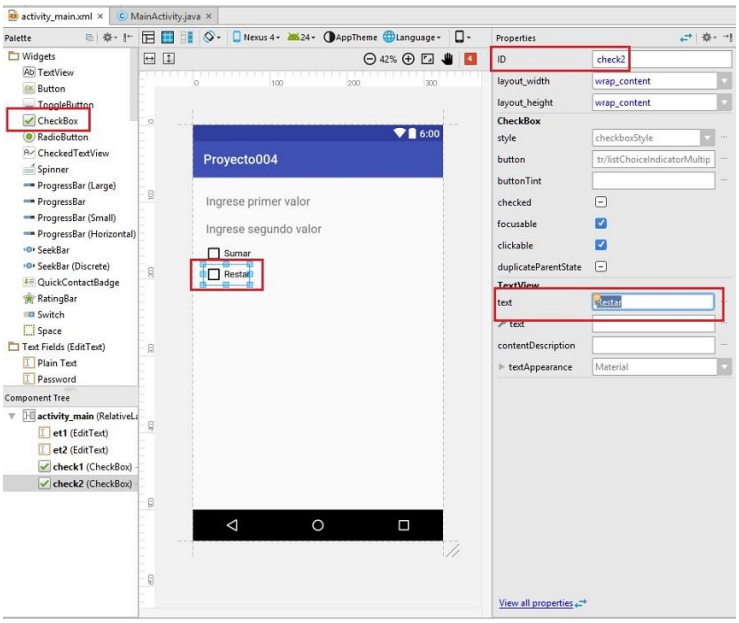
Control CheckBox

Ejecuta Android Studio y crea un nuevo proyecto. En Project Template selecciona Empty Activity y presiona Next.

Como nombre checkboxApp, en Package name escribe sv.edu.udb. checkboxapp en Save location escoge la carpeta de tu preferencia, en Language elige Java y el Minimum SDK selecciona API 16: Android 4.1 (Jelly Bean) y Presiona Finish.

Realizar la carga de dos números en controles de tipo EditText ("Number"). Mostrar en las propiedades "hint" de cada componente un mensaje que solicite la carga de los valores. Disponer dos controles de tipo CheckBox para seleccionar si queremos sumar y/o restar dichos valores. Finalmente, mediante un control de tipo Button efectuamos la operación respectiva. Mostramos el o los resultados en un TextView.

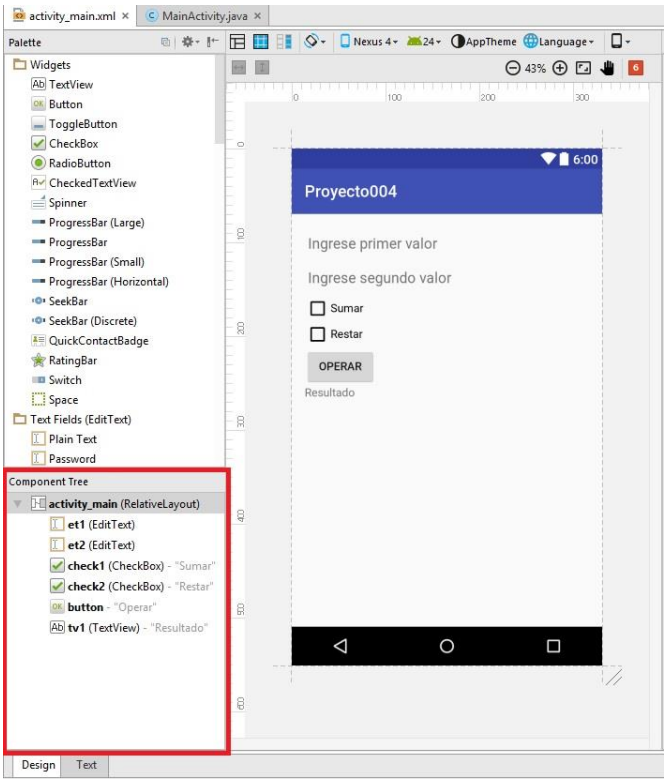
Lo nuevo en este problema es la inserción de dos objetos de la clase CheckBox que se encuentra en la pestaña "Widgets":



Debemos iniciar las propiedades "text" para mostrar un texto y la propiedad "id" para poder hacer referencia al CheckBox en el programa java.

El primer CheckBox definimos su "id" con el valor check1 y el segundo con el valor check2.

Luego la interfaz gráfica final para este problema y los nombres de los controles o componentes visuales los podemos ver en la ventana "Component Tree":



Controlar que fijamos los valores de las propiedades "id" de cada objeto: et1, et2, check1, check2 y tv1.

No olvidemos inicializar la propiedad onClick del objeto button con el valor "operar" (es el nombre del método a ejecutarse cuando se presione el botón y lo implementa la clase que hacemos) **Código fuente:**

```
package com.tutorialesprogramacionya.proyecto004;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.view.View;

import android.widget.CheckBox;

import android.widget.EditText;

import android.widget.TextView;


public class MainActivity extends AppCompatActivity {

  private EditText et1,  et2;

  private TextView tv1;
```

```
private CheckBox check1,  check2;

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

  et1 = (EditText) findViewById(R.id.et1);
  et2 = (EditText) findViewById(R.id.et2);
  tv1 = (TextView) findViewById(R.id.tv1);
  check1 = (CheckBox) findViewById(R.id.check1);
  check2 = (CheckBox) findViewById(R.id.check2);
}

//Este método se ejecutará cuando se presione el botón
public void operar(View view) {
  String valor1 = et1.getText().toString();
  String valor2 = et2.getText().toString();
  int nro1 = Integer.parseInt(valor1);
  int nro2 = Integer.parseInt(valor2);
  String resu = "";
  if (check1.isChecked() == true) {
    int suma = nro1 + nro2;
    resu = "La suma es: " + suma;
  }
  if (check2.isChecked() == true) {
    int resta = nro1 - nro2;
    resu = resu + " La resta es: " + resta;
  }
```

```
    tv1.setText(resu);
  }
}
```

Definimos dos objetos de la clase CheckBox como atributos de la clase:

```
    private CheckBox check1,check2;
```

En el método onCreate los inicializamos con los objetos definidos en el archivo XML:

```
        check1=(CheckBox)findViewById(R.id.check1);
        check2=(CheckBox)findViewById(R.id.check2);
```

En el método operar debemos definir dos if a la misma altura ya que los dos controles de tipo CheckBox pueden estar seleccionados simultáneamente. Definimos una variable de tipo String y la inicializamos con cadena vacía para el caso en que los dos CheckBox no estén seleccionados:

```
        String resu="";
        if (check1.isChecked()==true) {
           int suma=nro1+nro2;
           resu="La suma es: "+ suma;
        }
        if (check2.isChecked()==true) {
           int resta=nro1-nro2;
           resu=resu + " La resta es: "+ resta;
        }
        tv1.setText(resu);
```

Cuando ejecutamos el programa en el emulador tenemos como resultado:

Control Spinner

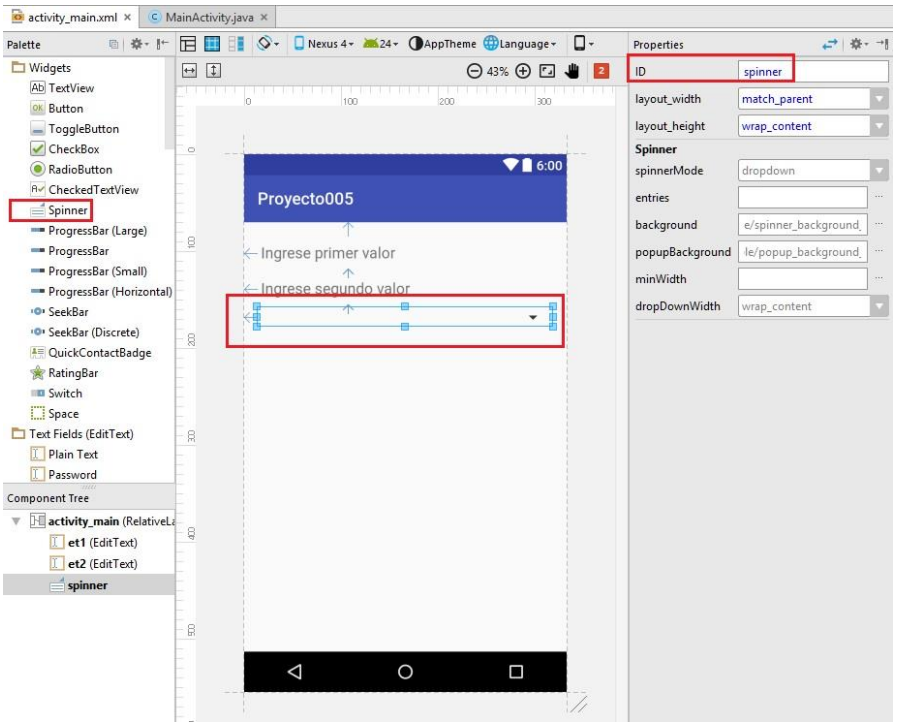
Ejecuta Android Studio y crea un nuevo proyecto. En Project Template selecciona Empty Activity y presiona Next.

Como nombre **spinnerApp**, en Package name escribe **sv.edu.udb.spinnerapp** en Save location escoge la carpeta de tu preferencia, en Language elige Java y el Minimum SDK selecciona API 16: Android 4.1 (Jelly Bean) y Presiona Finish.

El objetivo de este concepto es seguir practicando lo visto hasta ahora e incorporar el control visual Spinner. El control Spinner muestra una lista de String y nos permite seleccionar uno de ellos. Cuando se lo selecciona se abre y muestra todos sus elementos para permitir seleccionar uno de ellos.

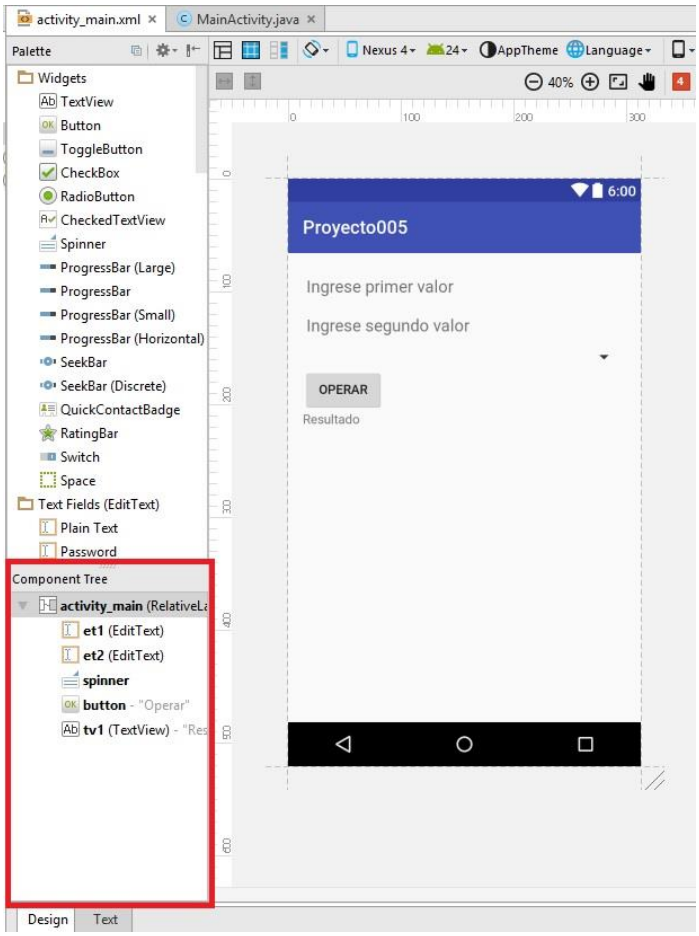
Realizar la carga de dos números en controles de tipo EditText ("Number"). Mostrar un mensaje que solicite la carga de los valores iniciando la propiedad "hint" de cada control. Disponer un control de tipo Spinner que permita seleccionar si queremos sumar, restar, multiplicar o dividir dichos valores. Finalmente, mediante un control de tipo Button efectuamos la operación respectiva. Mostramos el resultado en un TextView.

Lo nuevo en este problema es la inserción de un control de tipo Spinner que se encuentra en la pestaña "Widgets":



Dejamos la propiedad id con el valor spinner (valor por defecto que crea el Android Studio al insertar el objeto de la clase Spinner).

En la siguiente imagen en la ventana "Component Tree" del Android Studio podemos observar los objetos dispuestos en el formulario, sus id, sus textos y de que clase son cada uno:



No olvidemos inicializar la propiedad onClick del objeto button con el valor "operar" (dicho nombre es el método que debemos implementar) **Código fuente:**

```
import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.view.View;

import android.widget.ArrayAdapter;

import android.widget.EditText;

import android.widget.Spinner;

import android.widget.TextView;
```

```
public class MainActivity extends AppCompatActivity {
    private Spinner spinner1;
    private EditText et1,et2;
    private TextView tv1;

    public MainActivity() {
    }

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

        et1=(EditText)findViewById(R.id.et1);
        et2=(EditText)findViewById(R.id.et2);
        tv1=(TextView)findViewById(R.id.tv1);

        spinner1 = (Spinner) findViewById(R.id.spinner);
        String []opciones={"sumar","restar","multiplicar","dividir"};
        ArrayAdapter <String>adapter = new
ArrayAdapter<String>(this,android.R.layout.simple_spinner_item, opciones);
        spinner1.setAdapter(adapter);
    }
    //Este método se ejecutará cuando se presione el botón
    public void operar(View view) {
        String valor1=et1.getText().toString();
        String valor2=et2.getText().toString();
        int nro1=Integer.parseInt(valor1);
```

```
        int nro2=Integer.parseInt(valor2);
        String selec=spinner1.getSelectedItem().toString();
        if (selec.equals("sumar")) {
           int suma=nro1+nro2;
           String resu=String.valueOf(suma);
           tv1.setText(resu);
        } else
        if (selec.equals("restar")) {
           int resta=nro1-nro2;
           String resu=String.valueOf(resta);
           tv1.setText(resu);
        }
        else
        if (selec.equals("multiplicar")) {
           int multi=nro1*nro2;
           String resu=String.valueOf(multi);
           tv1.setText(resu);
        }
        else
        if (selec.equals("dividir")) {
           int divi=nro1/nro2;
           String resu=String.valueOf(divi);
           tv1.setText(resu);
        }
     }
}
```

Definimos un objeto de la clase Spinner:

```
private Spinner spinner1;
```

En el método onCreate obtenemos la referencia al control visual declarado en el archivo XML (Veamos que no es obligatorio que el nombre de nuestro objeto llamado spinner1 sea igual al id definido en la interfaz visual llamado spinner):

```
spinner1 = (Spinner) findViewById(R.id.spinner);
```

Definimos un vector con la lista de String que mostrará el Spinner:

```
String[]opciones={"sumar","restar","multiplicar","dividir"};
```

Definimos y creamos un objeto de la clase ArrayAdapter:

```
ArrayAdapter<String> adapter = new
ArrayAdapter<String>(this,android.R.layout.simple_spinner_item, opciones);
```

Al constructor le pasamos como primer parámetro la referencia de nuestro AppCompatActivity (this), el segundo parámetro indica el tipo de Spinner, pudiendo ser las constantes:

android.R.layout.simple_spinner_item

android.R.layout.simple_spinner_dropdown_item

El tercer parámetro es la referencia del vector que se mostrará.

Luego llamamos al método setAdapter de la clase Spinner pasando la referencia del objeto de la clase ArrayAdapter que acabamos de crear:

```
spinner1.setAdapter(adapter);
```

En el método operar que se ejecuta cuando presionamos el botón "OPERAR" y donde procedemos a extraer el contenido seleccionado del control Spinner:

```
String selec=spinner1.getSelectedItem().toString();
```

Luego mediante una serie de if anidados verificamos si debemos sumar, restar, multiplicar o dividir:

```
if (selec.equals("sumar")) {
   int suma=nro1+nro2;
   String resu=String.valueOf(suma);
   tv1.setText(resu);
} else
if (selec.equals("restar")) {
```

```
    int resta=nro1-nro2;
    String resu=String.valueOf(resta);
    tv1.setText(resu);
}
else
if (selec.equals("multiplicar")) {
    int multi=nro1*nro2;
    String resu=String.valueOf(multi);
    tv1.setText(resu);
}
else
if (selec.equals("dividir")) {
    int divi=nro1/nro2;
    String resu=String.valueOf(divi);
    tv1.setText(resu);
}
```

En el emulador tenemos como resultado de ejecutar el programa:

Si queremos que el Spinner no ocupe todo el ancho cambiamos la propiedad layout_width con el valor wrap_content.

Control ListView (con una lista de String)

Ejecuta Android Studio y crea un nuevo proyecto. En Project Template selecciona Empty Activity y presiona Next.

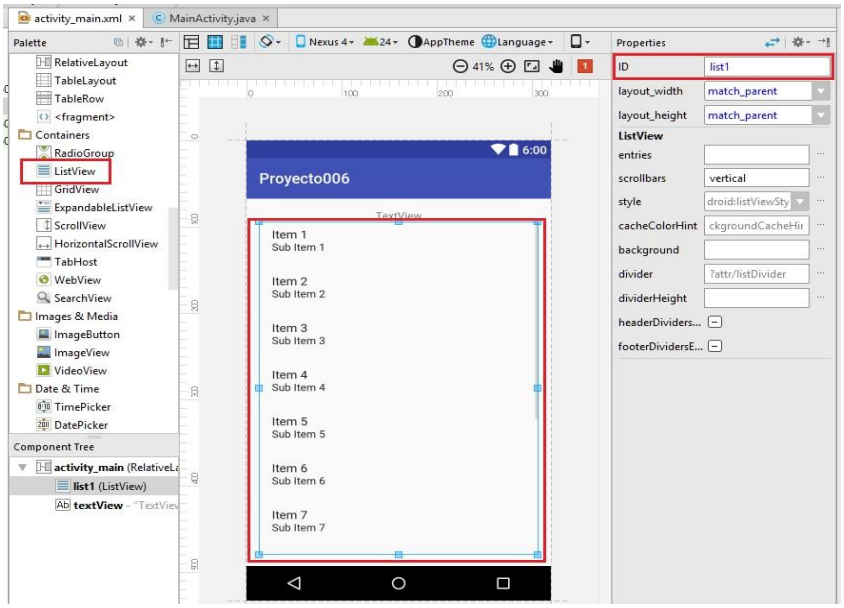
Como nombre ListView**App**, en Package name escribe **sv.edu.udb.**listview**app** en Save location escoge la carpeta de tu preferencia, en Language elige Java y el Minimum SDK selecciona API 16: Android 4.1 (Jelly Bean) y Presiona Finish.

El control ListView a diferencia del Spinner que se cierra luego de seleccionar un elemento permanecen visibles varios elementos (se lo utiliza cuando hay que mostrar muchos elementos)

En este primer ejemplo mostraremos una lista de String (es decir cada elemento de la lista es un String, veremos más adelante que podemos tener listas de objetos de otro tipo: imágenes, íconos, varios String por elemento etc.)

Si la lista no entra en el espacio que hemos fijado para el ListView nos permite hacer scroll de los mismos. El control ListView se encuentra en la pestaña "Containers".

Disponer un ListView con los nombres de paises de sudamérica. Cuando se seleccione un país mostrar en un TextView la cantidad de habitantes del país seleccionado.
La interfaz visual a implementar es la siguiente (primero disponemos un TextView en la parte superior (cuyo id lo definimos con el valor tv1) y un ListView (y definimos su id con el valor list1)):



Código fuente:

```
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {
   private String[] paises = { "Argentina", "Chile", "Paraguay", "Bolivia",
         "Peru", "Ecuador", "Brasil", "Colombia", "Venezuela", "Uruguay" };

   private String[] habitantes = { "40000000", "17000000", "6500000",
         "10000000", "30000000", "14000000", "183000000", "44000000",
         "29000000", "3500000" };

   private TextView tv1;
   private ListView lv1;

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

      tv1=(TextView)findViewById(R.id.tv1);
      lv1 =(ListView)findViewById(R.id.list1);
```

```
        ArrayAdapter <String>adapter = new

ArrayAdapter<String>(this,android.R.layout.simple_list_item_1, paises);

        lv1.setAdapter(adapter);

        lv1.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView adapterView, View view, int i, long l) {
                tv1.setText("Población de "+ lv1.getItemAtPosition(i) + " es "+ habitantes[i]);
            }
        });
    }
}
```

Primero definimos dos vectores paralelos donde almacenamos en uno los nombres de países y en el otro almacenamos la cantidad de habitantes de dichos países:

```
private
String[]paises={"Argentina","Chile","Paraguay","Bolivia","Peru","Ecuador","Brasil","Colom
bia","Venezuela","Uruguay"};

private String[]habitantes=
{"40000000","17000000","6500000","10000000","30000000","14000000","183000000","
44000000","29000000","3500000"};
```

Definimos un objeto de tipo TextView y otro de tipo ListView donde almacenaremos las referencias a los objetos que definimos en el archivo XML:

```
    private TextView tv1;
    private ListView lv1;
```

En el método onCreate obtenemos la referencia a los dos objetos:

```
tv1=(TextView)findViewById(R.id.tv1);
lv1 =(ListView)findViewById(R.id.list1);
```

Creamos un objeto de la clase ArrayAdapter de forma similar a como lo hicimos cuando vimos la clase Spinner:

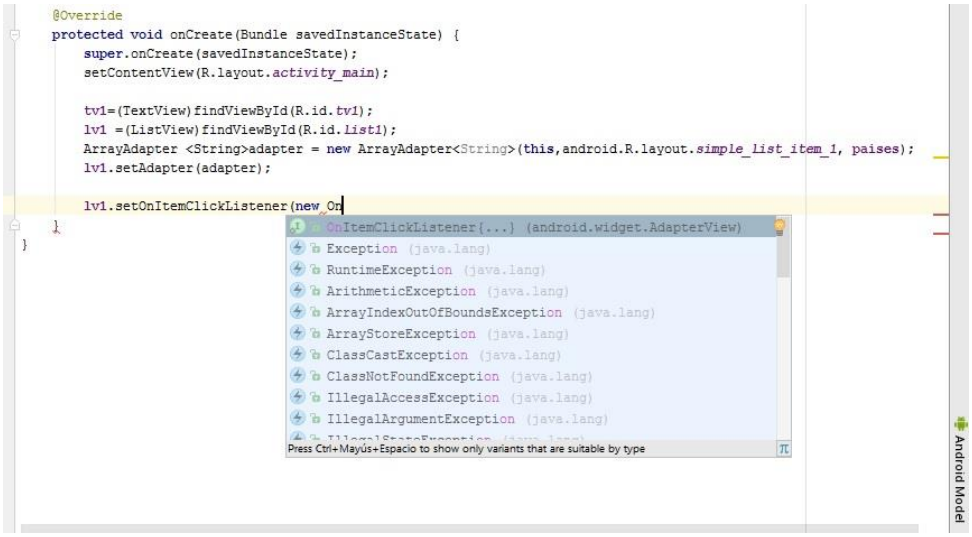
```
ArrayAdapter<String> adapter = new
ArrayAdapter<String>(this,android.R.layout.simple_list_item_1, paises);

lv1.setAdapter(adapter);
```

Llamamos al método setOnItemClicListener de la clase ListView y le pasamos como parámetro una clase anónima que implementa la interfaz OnItemClickListener (dicha interfaz define el método onItemClick que se dispara cuando seleccionamos un elemento del ListView):

```
lv1.setOnItemClickListener(new AdapterView.OnItemClickListener() {
        @Override
        public void onItemClick(AdapterView adapterView, View view, int i, long l) {
           tv1.setText("Población de "+ lv1.getItemAtPosition(i) + " es "+ habitantes[i]);
        }
    });
```

Para codificar esta clase anónima podemos utilizar la facilidad que nos brinda el Android Studio para generarla automáticamente, para esto tipeamos:



Luego presionamos la tecla "Enter" y el Android Studio automáticamente nos genera la clase anónima implementando la interfaz onItemClick.

Dentro del método onItemClick modificamos el contenido del TextView con el nombre del país y la cantidad de habitantes de dicho país. Este método recibe en el tercer parámetro la posición del item seleccionado del ListView.

Cuando ejecutamos el proyecto podemos ver una interfaz en el emulador similar a esta: