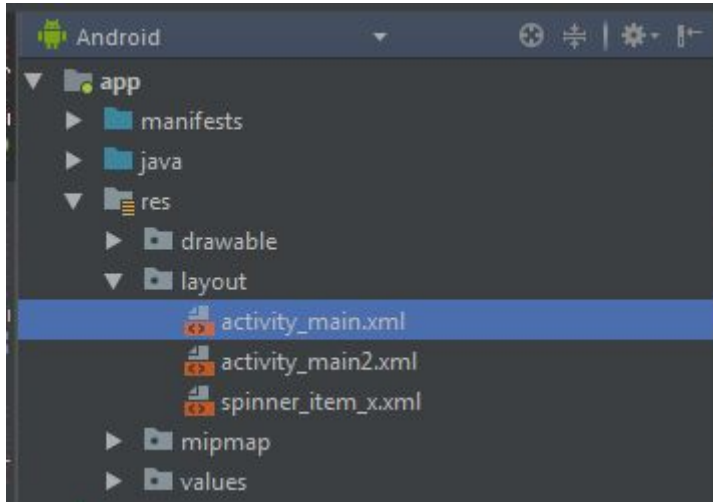


<b>Interfaz Gráfica</b>	<b>2</b>
<b>Declaración de componentes</b>	<b>4</b>
<b>Adaptador</b>	<b>6</b>
<b>Métodos para ejecutarse en eventos</b>	<b>7</b>

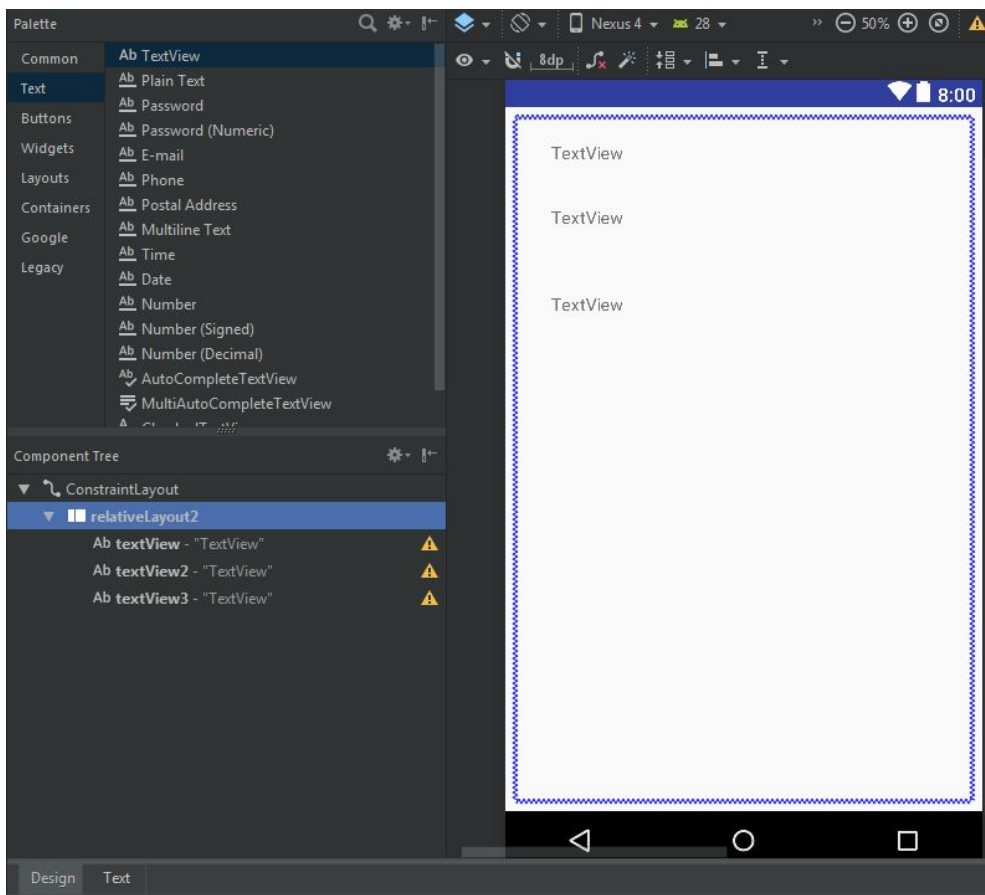
# Interfaz Gráfica

en la estructura de la aplicación los archivos `activity_main.xml` componen la interfaz gráfica



en ella podemos encontrar 2 tipos de vistas:

## Vista Diseño:

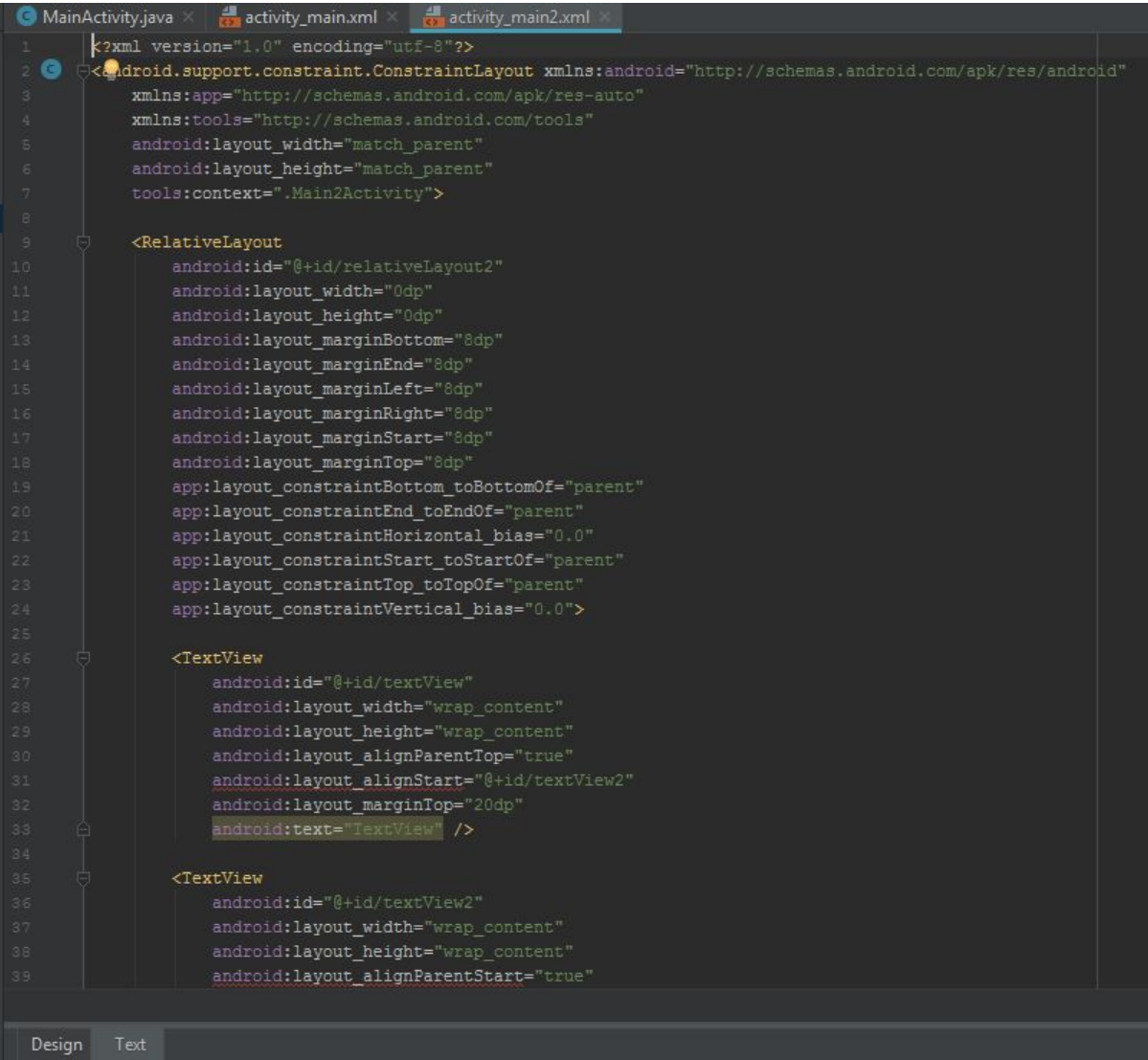


en esta vista podemos arrastrar componentes y ubicarlos de la manera que requiera, antes de colocar cualquier componente, debemos agregar el layout, en este caso RELATIVE

LAYOUT el cual nos permite ubicar los componentes con mayor libertad. todos los demás componentes (EditText, Spinner, Textview) deben ir dentro de este LAYOUT.

### Vista Texto

en esta vista podemos encontrar el código que contiene toda la información de los componentes, es posible editarlos si así lo deseamos.

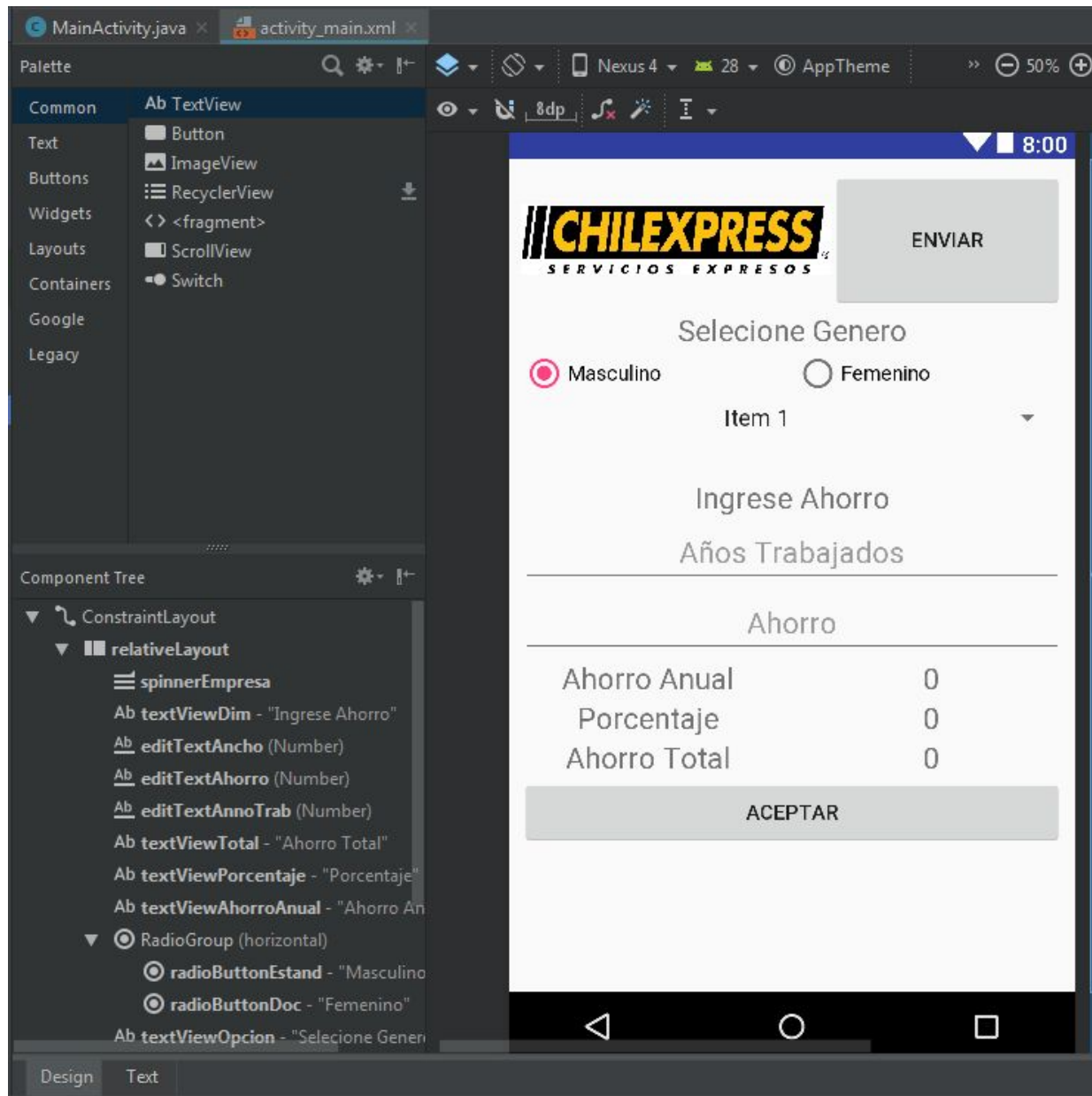


```
1  <?xml version="1.0" encoding="utf-8"?>
2  <android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:app="http://schemas.android.com/apk/res-auto"
4      xmlns:tools="http://schemas.android.com/tools"
5      android:layout_width="match_parent"
6      android:layout_height="match_parent"
7      tools:context=".Main2Activity">
8
9      <RelativeLayout
10         android:id="@+id/relativeLayout2"
11         android:layout_width="0dp"
12         android:layout_height="0dp"
13         android:layout_marginBottom="8dp"
14         android:layout_marginEnd="8dp"
15         android:layout_marginLeft="8dp"
16         android:layout_marginRight="8dp"
17         android:layout_marginStart="8dp"
18         android:layout_marginTop="8dp"
19         app:layout_constraintBottom_toBottomOf="parent"
20         app:layout_constraintEnd_toEndOf="parent"
21         app:layout_constraintHorizontal_bias="0.0"
22         app:layout_constraintStart_toStartOf="parent"
23         app:layout_constraintTop_toTopOf="parent"
24         app:layout_constraintVertical_bias="0.0">
25
26         <TextView
27             android:id="@+id/textView"
28             android:layout_width="wrap_content"
29             android:layout_height="wrap_content"
30             android:layout_alignParentTop="true"
31             android:layout_alignStart="@+id/textView2"
32             android:layout_marginTop="20dp"
33             android:text="TextView" />
34
35         <TextView
36             android:id="@+id/textView2"
37             android:layout_width="wrap_content"
38             android:layout_height="wrap_content"
39             android:layout_alignParentStart="true"
```

sin embargo nos centraremos en la **Vista Diseño**.

# Declaración de componentes

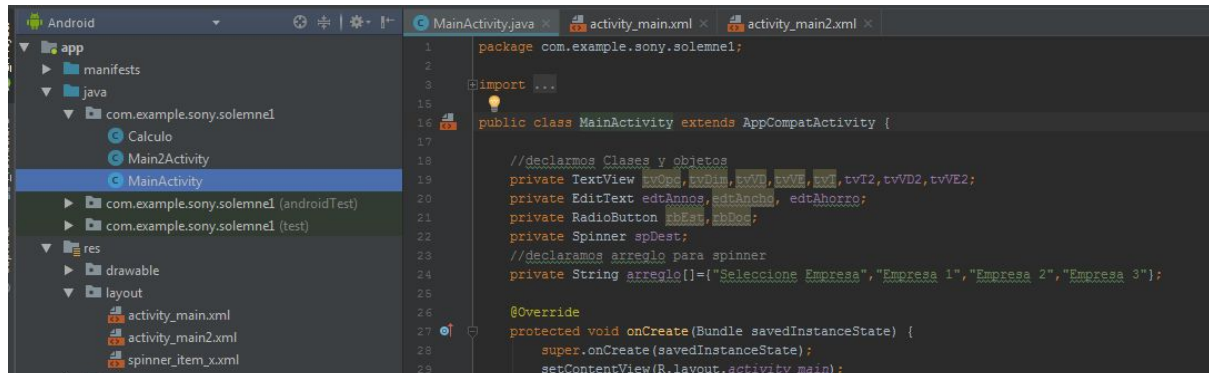
cada componente que agregamos a nuestra interfaz debe ser declarado en nuestra clase principal.



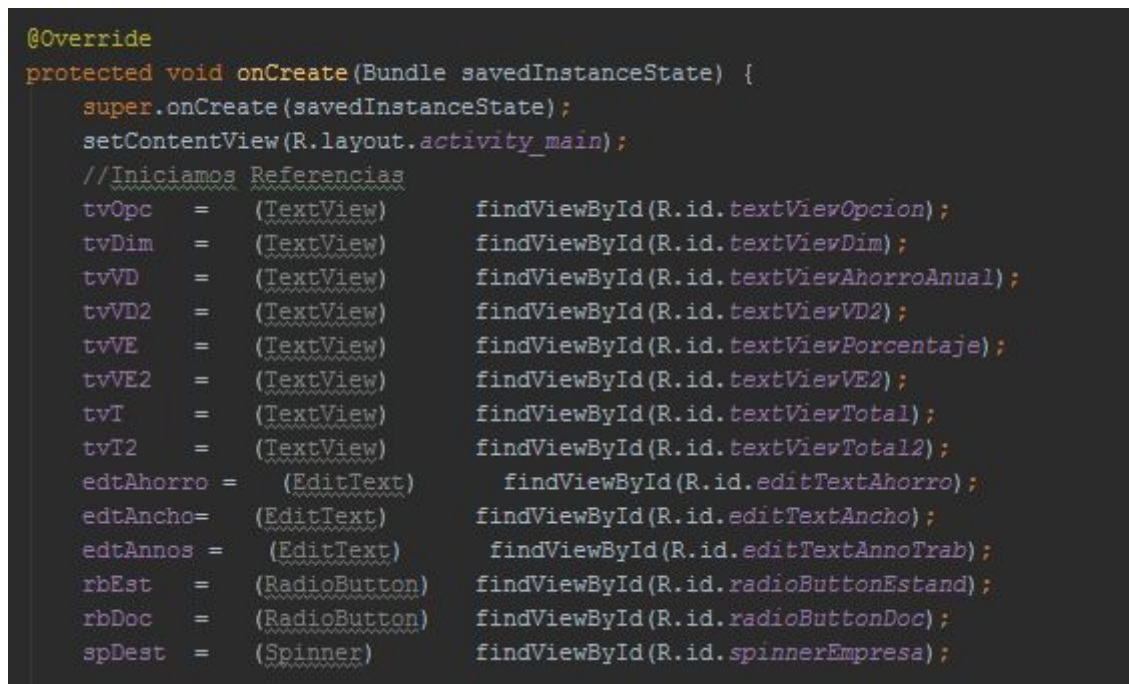
en este caso declararemos todos los componentes como si se trataran de objetos, esto lo realizamos antes de el método onCreate y dentro de la clase MainActivity.

nota:

**TextView** seria la clase y **IvOpc** el objeto segun este ejemplo.



.luego dentro del método onCreate Iniciaremos las referencias.



en el ejemplo de esta se realizan un casteo del componente a objeto, con el método **findViewById()** donde **R.id.id\_del\_componente** es donde se coloca el ID del componente

id	textViewOpcion
layout_width	match_parent
layout_height	wrap_content
Layout_Margin	[?, ?, ?, ?]
Padding	[?, ?, ?, ?]
Theme	
fontFamily	sans-serif
layout_below	@+id/imageView2
text	Seleccione Genero
textColorHighli	@color/colorPrime
textSize	20dp
accessibilityHe	
accessibilityPar	
alpha	

# Adaptador

El adaptador se utiliza para llenar de datos un **SPINNER** o **LISTVIEW** con un arreglo, para ellos debemos declarar el **arreglo**.

```
public class MainActivity extends AppCompatActivity {  
  
    //declaramos Clases y objetos  
    private TextView tvOpc, tvDim, tvVD, tvVE, tvT, tvT2, tvVD2, tvVE2;  
    private EditText edtAnnos, edtAncho, edtAhorro;  
    private RadioButton rbEst, rbDoc;  
    private Spinner spDest;  
    //declaramos arreglo para spinner  
    private String arreglo[]={"Seleccione Empresa", "Empresa 1", "Empresa 2", "Empresa 3"};
```

este puede ser llenado con valores iniciales o puede ser llenado más adelante en el método onCreate dependiendo del caso.

## DENTRO DEL MÉTODO ONCREATE:

```
/*ciclo para llenar Spinner (no aplica en este caso)  
for(int i=0;i<arreglo.length;i++){  
  
}  
*/  
  
//delaramos adaptador entre Array y Spinner  
ArrayAdapter<String>adaptador;  
adaptador=new ArrayAdapter<String>( context: this,R.layout.spinner_item_x,arreglo);  
spDest.setAdapter(adaptador);
```

como se aprecia, existe un código comentado, el cual contenía un ciclo para llenar el arreglo que poblaría el spinner, sin embargo en este caso no es necesario.

Más abajo se ve la declaración de un adaptador y dentro del constructor del ArrayAdapter se ingresan los argumentos: "this" que indica el contexto, es decir esta clase, R.layout es la forma en que se visualizará el spinner, stylos, fuentes ,color etc, y finalmente el arreglo de donde provienen los datos.

luego el objeto spDest en este caso corresponde al spinner y esta utilizando el método setAdapter, dentro de este se coloca el adaptador y con este sera posible por fin poblar con datos los componentes.

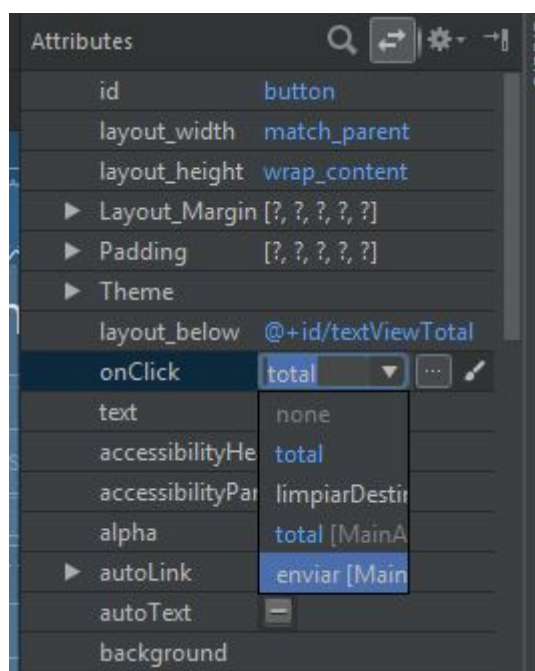


## Métodos para ejecutarse en eventos

Fuera del método onCreate podemos Crear métodos para ser ejecutados desde un evento, para ellos debemos crear el método y como argumentos colocar View view.

```
public void total(View view){
    //declaramos variables;
    int annos=0,ahorro=0,alto=0;
    double desc=0;
    //extraemos contenido de los textbox
    if(edtAnnos.getText().toString().trim().equalsIgnoreCase(" ")){
        edtAnnos.setError("Ingrese años trabajados");
    }else if(edtAhorro.getText().toString().trim().equalsIgnoreCase(" ")){
        edtAhorro.setError("Ingrese Ahorro");
    }
}
```

con lo anterior, el atributo onclick mostrará el método a ejecutar, como su nombre lo dice al hacer click.



existen componentes los para los cuales **NO son válidos** los eventos **onclick** por ejemplo el **SPINNER** y el **LISTVIEW**, los cuales contienen lista de items. Para ellos, debemos utilizar métodos de las librerías que permiten simular onclick en los ítems de estos.

A diferencia de los anteriores, estos métodos heredados van **dentro** del método **oncreate**.

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

    //evaluamos el indice de nuestro spinner
    int posicion = spDest.getSelectedItemPosition();

    //instanciamos la clase calculo para realizar la logica
    Calculo calculo = new Calculo();

    double valor = calculo.ganancia(posicion);

    //asignamos un valor inicial al textview
    tvI2.setText("0");
    //mostramos el valor en textview
    tvVE2.setText(String.valueOf((valor*100)));

}

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

}

});
```

Solemne1

**CHILEXPRESS**  
SERVICIOS EXPRESOS

ENVIAR

Seleccione Genero

☒ Masculino ☐ Femenino

Seleccione Empresa

Empresa 1

**Empresa 2**

Empresa 3

Ahorro

Ahorro Anual	0
Porcentaje	0.0
Ahorro Total	0

ACEPTAR

Solemne1

**CHILEXPRESS**  
SERVICIOS EXPRESOS

ENVIAR

Seleccione Genero

☒ Masculino ☐ Femenino

Empresa 2

Ingrese Ahorro

Años Trabajados

Ahorro

Ahorro Anual	0
Porcentaje	2.5
Ahorro Total	0

ACEPTAR