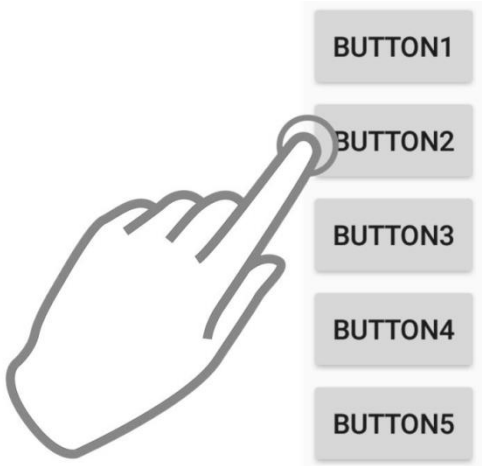
Gestión de eventos

Lógica
de
vistas básicas

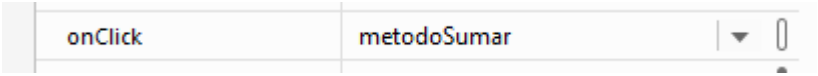
Eventos

- Cuando el **usuario interacciona con la interfaz** y pulsa un botón (por ejemplo) se dice que ha generado un evento.
- La **lógica de la aplicación debe especificar qué acciones** se deben llevar a cabo cuando se producen determinados eventos.
- Puede hacerse con **métodos propios o implementados**



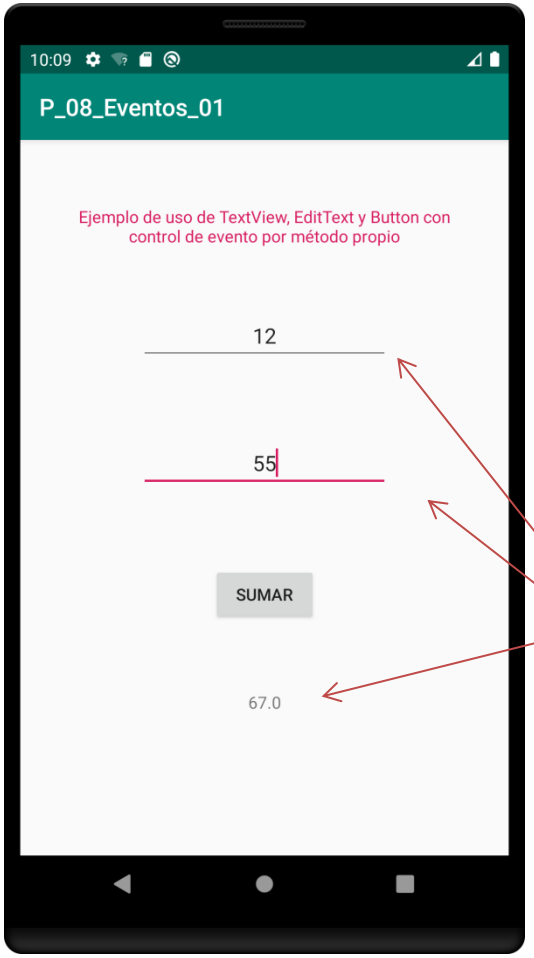
Método propio: Atributo onClick

- El atributo android:onClick de una vista nos permite asignar **desde un layout XML un método propio** que será ejecutado al hacer click en una vista. Resulta muy sencillo y además está disponible en cualquier descendiente de la clase View (botones, imágenes,…).



- Sin embargo esta técnica presenta inconvenientes:
  - solo está disponible para el evento onClick(). La clase View tiene otros eventos (onLongClick(), onFocusChange(), onKey(),…) para los que no se han definido un atributo xml.
  - además, estamos mezclando lógica y diseño lo que está desaconsejado en programación.

P_08_Eventos_1
Vistas EditText - TextView – Button
Gestión de evento por método propio




```
package com.pdm.p_08_eventos_1;

import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {

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

  public void metodoSumar(View view) {
    EditText editText1 = findViewById(R.id.editText);
    EditText editText2 = findViewById(R.id.editText2);
    TextView textView = findViewById(R.id.textView2);
    String valor1 = editText1.getText().toString();
    double num1 = Double.parseDouble(valor1);
    String valor2 = editText2.getText().toString();
    double num2 = Double.parseDouble(valor2);
    double suma = num1 + num2;
    String resultado = String.valueOf(suma);
    textView.setText(resultado);
  }
}
```

- En el atributo onClick de la vista Button se ha asignado el método propio metodoSumar que debe recibir como parámetro el valor de la vista view (aquella sobre la que se ha hecho "click“). onClick metodoSumar
- Para acceder a los elementos del layout, utilizamos el método ***findViewById()*** (explicado en 19_Diseño_y_lógica).
- Para obtener los valores tecleados en los EditText se llama al método ***getText()*** y tras los cálculos necesarios se coloca el resultado en el TextView con el método ***setText(CharSequence text)***.

Gestión de eventos por métodos implementados

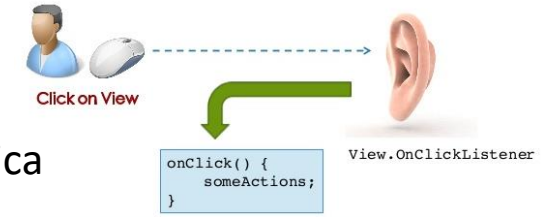
- Android recomienda tratar los eventos mediante:
  - **Event Listener (detector de eventos)**: escuchan eventos generados en una View o ViewGroup. Cada Listener tiene solo un método que será llamado cuando el usuario interactúa con la vista. Ej onClick, onLongClick, onFocusChanged, etc…
  - **Event Handler (manejador de eventos)**: maneja los eventos de entrada sin importar donde está el foco. No están necesariamente asociados a una vista. Ej: pulsar el Botón atrás, tocar la pantalla…

Listener

- Un EventListener **detecta** ("escucha") **eventos** generados por una vista (el botón en nuestro caso anterior).
- Para cada vista que pueda generar un evento (un Button o un EditText,…) **es posible especificar a qué tipo de detector** se le notificarán los eventos que produzca.
- Para ello es necesario registrarlo mediante el método **setOnXXXListener** apropiado:
  - setOnClickListener
  - setOnLongClickListener
  - etc.

Tipos de Listener

- La clase **View** define interfaces que deben cumplir los diferentes tipos de *Listeners:*
  - **View.OnClickListener**: Para procesar notificaciones de click
  - **View.OnLongClickListener**: Para procesar las notificaciones de click mantenido
  - **View.OnFocusChangeListener**: Para procesar la adquisición o pérdida del foco
  - **View.OnKeyListener**: Para procesar las pulsaciones de teclas físicas del dispositivo. Sólo se reciben estas notificaciones si el elemento tiene el foco mientras se realiza la pulsación.
  - **View.OnTouchListener**: Cuando el usuario toca, pulsando o liberando, un elemento.
  - **View.OnCreateContextMenuListener**: Cuando se crea un menú contextual porque el usuario ha mantenido una pulsación larga
  - …
- Cada una de estas interfaces declara un **método** (onClick(), onKey (),…) que es **llamado cuando se recibe el evento**. Al implementar dicho método se especifica cómo se **gestiona el evento**.



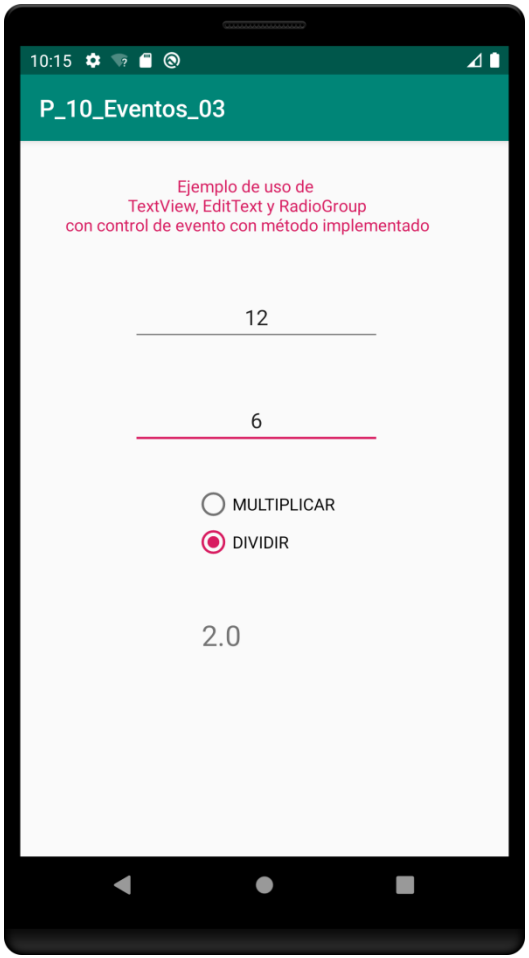
- La clase View también dispone de los correspondientes métodos setOn<accion>Listener() para **registrar** los listeners para cada tipo de acción.

P_09_Eventos_02
OnClickListener - setOnClickListener - onClick


```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Button button = findViewById(R.id.button);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                EditText editText1 = findViewById(R.id.editText);
                EditText editText2 = findViewById(R.id.editText2);
                TextView textView = findViewById(R.id.textView2);
                String valor1 = editText1.getText().toString();
                double num1 = Double.parseDouble(valor1);
                String valor2 = editText2.getText().toString();
                double num2 = Double.parseDouble(valor2);
                double divi= num1 / num2;
                String resultado = String.valueOf(divi);
                textView.setText(resultado);
            }
        });
    }

}
```

Al botón le asociamos un detector de eventos "click" con el método ***setOnClickListener*** de tal manera que crea una instancia (new ***View.OnClickListener***) que llamará al método ***onClick*** que siempre debe ser **public void y recibir como parámetro un objeto de la clase View** (en nuestro caso AS lo ha llamado view) y que es la vista sobre la que se ha "clicado" (en nuestro caso el botón)

P_10_Eventos_03
Vistas RadioGroup y RadioButton
OnCheckedChangedListener - setOnCheckedListener - onCheckedChanged

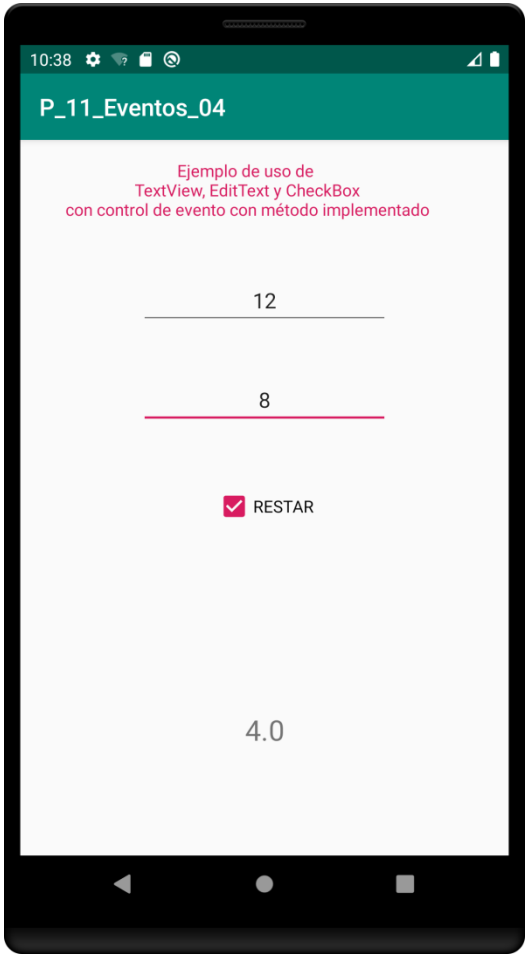



```
public class MainActivity extends AppCompatActivity {

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    RadioGroup radioGroup= findViewById(R.id.radioGroup);
    radioGroup.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {
      @Override
      public void onCheckedChanged(RadioGroup group,  int i) {
        EditText editText1 = findViewById(R.id.editText1);
        EditText editText2 = findViewById(R.id.editText2);
        RadioButton radioButton=findViewById(R.id.radioButton);
        TextView textView = findViewById(R.id.textView2);
        String valor1 = editText1.getText().toString();
        double num1 = Double.parseDouble(valor1);
        String valor2 = editText2.getText().toString();
        double num2 = Double.parseDouble(valor2);
        double operacion = 0;
        if (radioButton.isChecked())
          operacion=num1*num2;
        if (i==R.id.radioButton2)
          operacion=num1/num2;
        String resultado = String.valueOf(operacion);
        textView.setText(resultado);
      }
    });
  }
}
```

Podemos trabajar con el identificador de la vista (radioButton) previamente definido llamando al método **isChecked**

O trabajar con el valor recibido como parámetro **i** que es el identificador del RadioButton seleccionado.

P_11_Eventos_04
Vista CheckBox



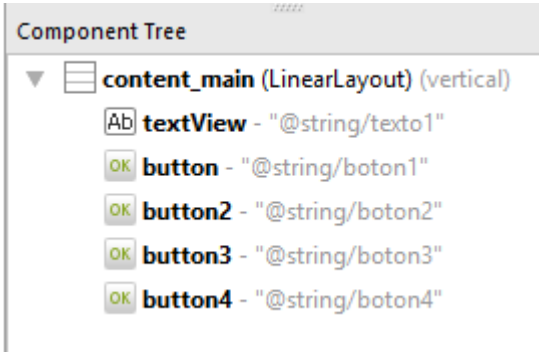

```
public class MainActivity extends AppCompatActivity {

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    CheckBox checkBox = findViewById(R.id.checkBox);
    checkBox.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
      @Override
      public void onCheckedChanged(CompoundButton buttonView, boolean b) {
        EditText editText1 = findViewById(R.id.editText);
        EditText editText2 = findViewById(R.id.editText2);
        TextView textView = findViewById(R.id.textView2);
        String valor1 = editText1.getText().toString();
        double num1 = Double.parseDouble(valor1);
        String valor2 = editText2.getText().toString();
        double num2 = Double.parseDouble(valor2);
        String resultado;
        if (b) {
          double operacion = num1 - num2;
          resultado = String.valueOf(operacion);
        } else
          resultado = "Active CheckBox para ver resultado";
        textView.setText(resultado);
      }
    });
  }
}
```

CompoundButton: botón que solo tiene dos estados posibles

Optimizando el Listener

- En numerosas ocasiones nuestras aplicaciones tienen varias vistas del mismo tipo "escuchando" eventos y se desea implementar un único método que sirva para todos ellas.
- **Se recomienda que sea la actividad la que implemente la interfaz onXXXListener y no cada una de las vistas receptoras de eventos.**
- Si existen varias vistas que deben detectar el mismo evento (por ejemplo OnClickListener) bastará con programar un método (onClick) y dentro de él podremos discriminar las distintas acciones a realizar dependiendo del valor del parámetro recibido .

Ejemplo de optimización del Listener




```
public class MainActivity extends AppCompatActivity implements View.OnClickListener {

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    ...
    Button button=findViewById(R.id.button);
    Button button2=findViewById(R.id.button2);
    Button button3=findViewById(R.id.button3);
    Button button4=findViewById(R.id.button4);
    button.setOnClickListener(this);
    button2.setOnClickListener(this);
    button3.setOnClickListener(this);
    button4.setOnClickListener(this);
  }

  @Override
  public void onClick(View view) {
    switch (view.getId()) {
      case R.id.button:
        //acciones para botón1
        break;
      case R.id.button2:
        //acciones para botón2
        break;
      case R.id.button3:
        //acciones para botón3;
        break;
      default:
        //acciones para resto de botones;
    }
  }
```

Qué es this?
El contexto de la aplicación!

De la vista "clickada" obtenemos su identificador con el método **getId()**.

Contexto de la aplicación

- Cuando se compila una aplicación, se crea una clase llamada R que contiene referencias a los recursos de la aplicación.
- El archivo de manifiesto de aplicación y estos recursos se combinan para crear lo que se conoce como el contexto de aplicación.
- Este contexto, representado por la clase Context, se puede utilizar en el código de la aplicación para obtener acceso a los recursos de la aplicación en tiempo de ejecución y para comunicarse con los otros componentes de la aplicación.
- Además, la clase Context cuenta con una amplia gama de métodos que pueden ser llamados para recopilar información y realizar cambios en las variables de entorno de la aplicación.
- Buena explicación

P_12_Eventos_05
Vista ToggleButton


```
public class MainActivity extends AppCompatActivity implements View.OnClickListener {

    private ToggleButton toggleButton;
    private TextView textView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        toggleButton = findViewById(R.id.toggleButton);
        textView = findViewById(R.id.textView2);
        toggleButton.setOnClickListener(this);

    }

    @Override
    public void onClick(View view) {
        if (toggleButton.isChecked())
            textView.setText(R.string.textoon);
        else
            textView.setText(R.string.textooff);

    }
}
```

Variables globales (o campos según AS) para que sean accesibles en todos los métodos

Hacemos uso de strings definidas como recursos

Prácticas

- Realiza los ejercicios propuestos en Ejercicios_03_Eventos.