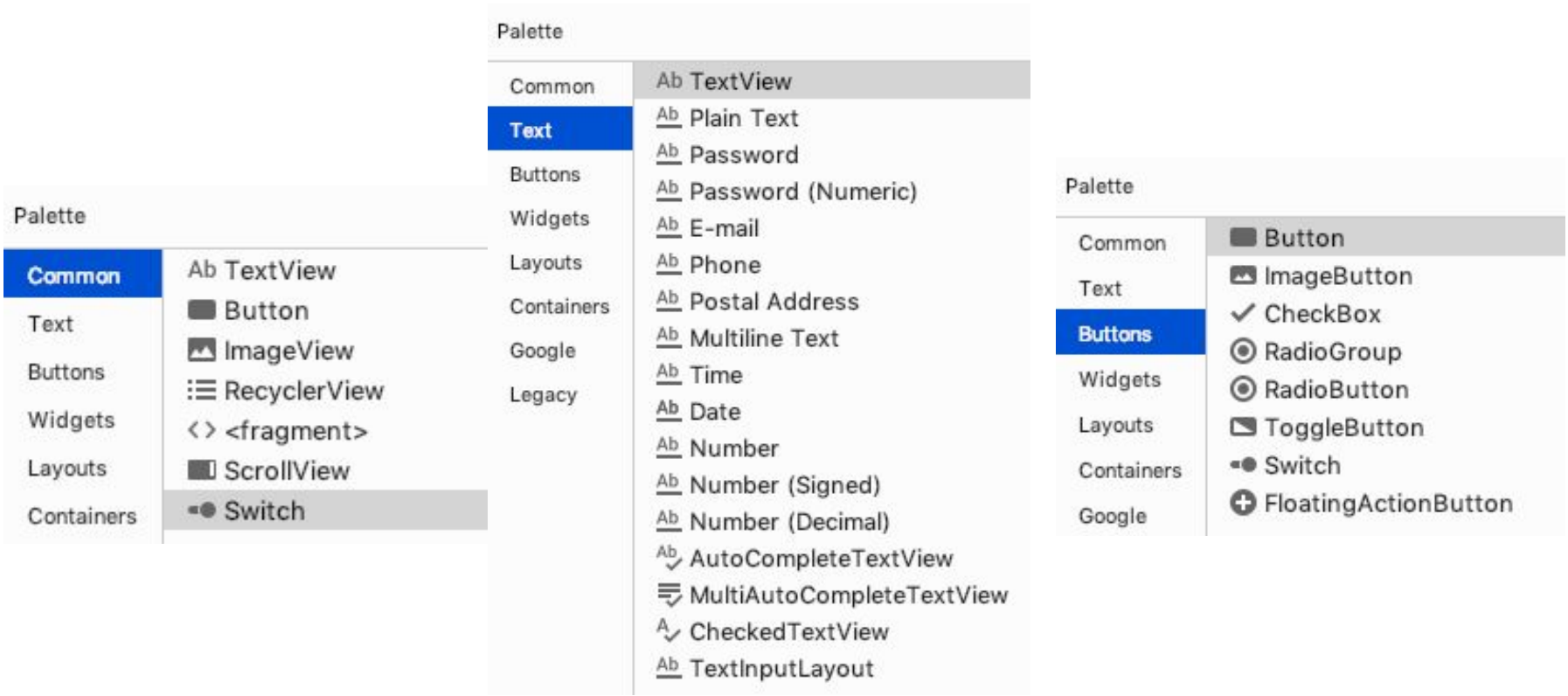
Apartado 8:
Programación con widgets.

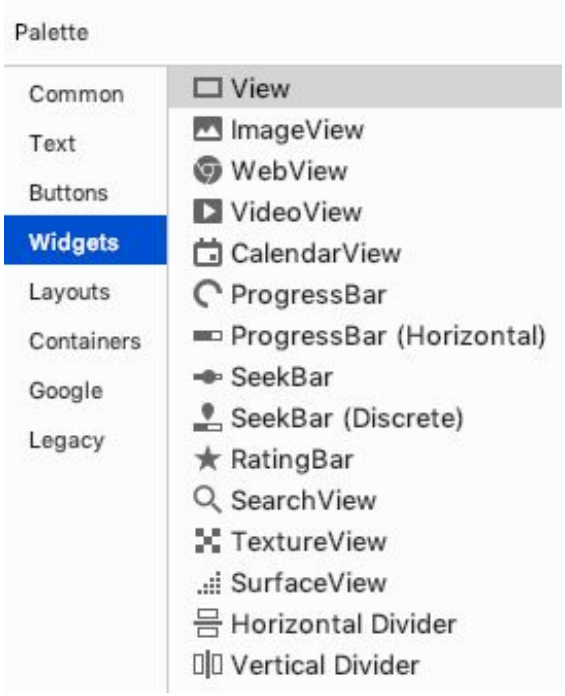
Autores:
Víctor M. Rivas Santos / Miguel Á. García Cumbreras
(Antonio Rueda Ruiz)

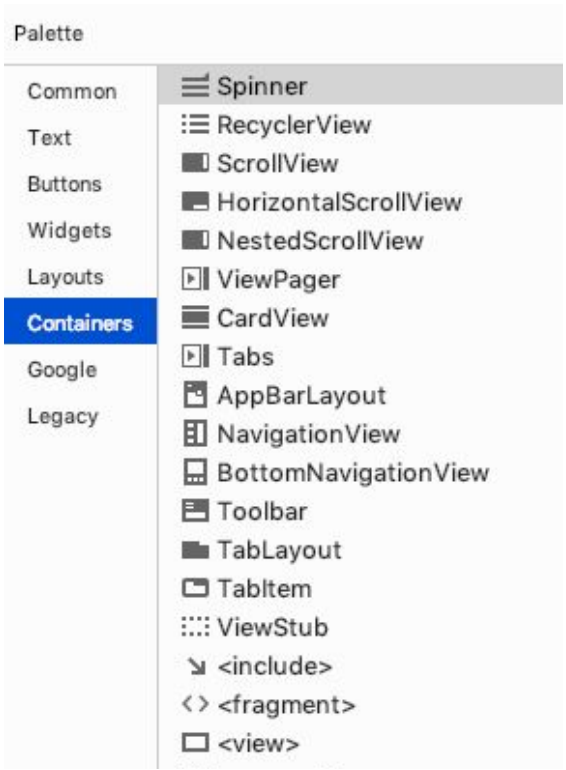
Definición

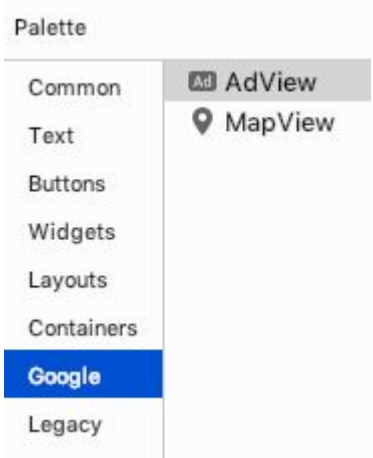
- Las *views* o *widgets* son cada uno de los elementos que podemos incluir en nuestra interfaz



Definición (II)







Identificación en fichero XML

- Cada *widget* debe tener un identificador único

android:id="@+id/btAceptar"

  - El signo “+” hace que se añada a la clase *R*

```
<Button
  android:text="Aceptar"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:id="@+id/btAceptar" />
```

Identificación en fichero Java (II)

- Una vez instanciado en Java, se podrá acceder a sus métodos.

```
package com.example.vrivas.testwidgets;

import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {
  TextView texto;
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    texto=(TextView) findViewById(R.id.txtMensajito);
    Log.d("MyApp", (String) texto.getText());
    texto.setText("No entiendo lo que dices :) ");
  }
}
```

Creación dinámica de widgets

- En tiempo de ejecución, es posible crear los widgets que necesitemos

```
public class MainActivity extends AppCompatActivity {
  Button btOK;
  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    btOK=new Button(this);
    btOK.setText("OKi");
    RelativeLayout.LayoutParams params = new RelativeLayout.LayoutParams(100,200);
    this.addContentView(btOK, params);

  }
}
```

EJERCICIO

5 widgets al día

MTV: Conocer algunas de las propiedades de los widgets

- Realizar una app que incluya (al menos) los siguientes widgets y aplicar mediante Java los cambios que se indican
  - *TextView*: cambiar el texto inicial que pone por defecto por cualquier otro.
  - *Button*: deshabilitarlo para que no pueda pincharse sobre él.
  - *ToggleButton*: deberá alternar entre los valores “Encendido” y “Apagado” al pinchar en él.
  - *Phone*, con valor inicial “12345678”: se deberá marcar como texto seleccionado la secuencia “456”
  - *ProgressBar* cuyo valor máximo sea 100: se deberá establecer su propiedad *progress* a 80.
- Sube a la plataforma el proyecto completo excluyendo el contenido del directorio *build*

Captura de eventos

- Los *widgets* posibilitan la interacción con el usuario
  - El código debe estar preparado para *responder* a los eventos que se producen
  - Un método que se invoca cuando se desea procesar un evento se denomina *callback*
- La captura y procesamiento de eventos se realiza usando **interfaces** …
  - … que a su vez implementan la interfaz *EventListener*

Algunos eventos (I)

1. View Event Listeners

- setOnClickListener - Callback when the view is clicked
- setOnDragListener - Callback when the view is dragged
- setOnFocusChangeListener - Callback when the view changes focus
- setOnGenericMotionListener - Callback for arbitrary gestures
- setOnHoverListener - Callback for hovering over the view
- setOnKeyListener - Callback for pressing a hardware key when view has focus
- setOnLongClickListener - Callback for pressing and holding a view
- setOnTouchListener - Callback for touching down or up on a view
- y muchos más: https://developer.android.com/guide/topics/ui/ui-events

Algunos eventos (II)

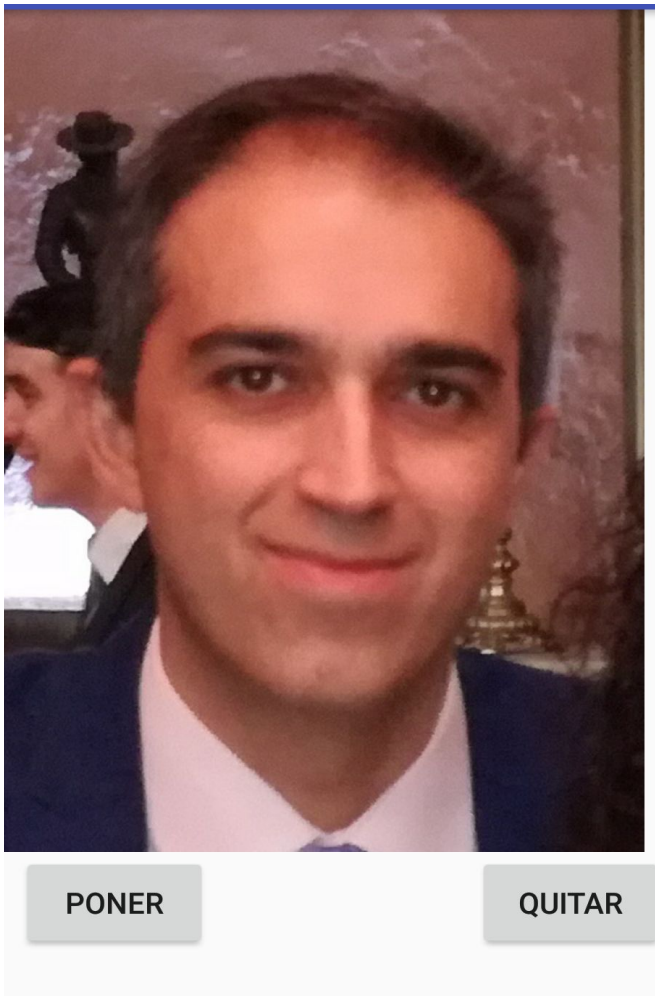
2. EditText Common Listeners

- addTextChangedListener - Fires each time the text in the field is being changed
- setOnEditorActionListener - Fires when an "action" button on the soft keyboard is pressed
- y muchos más...

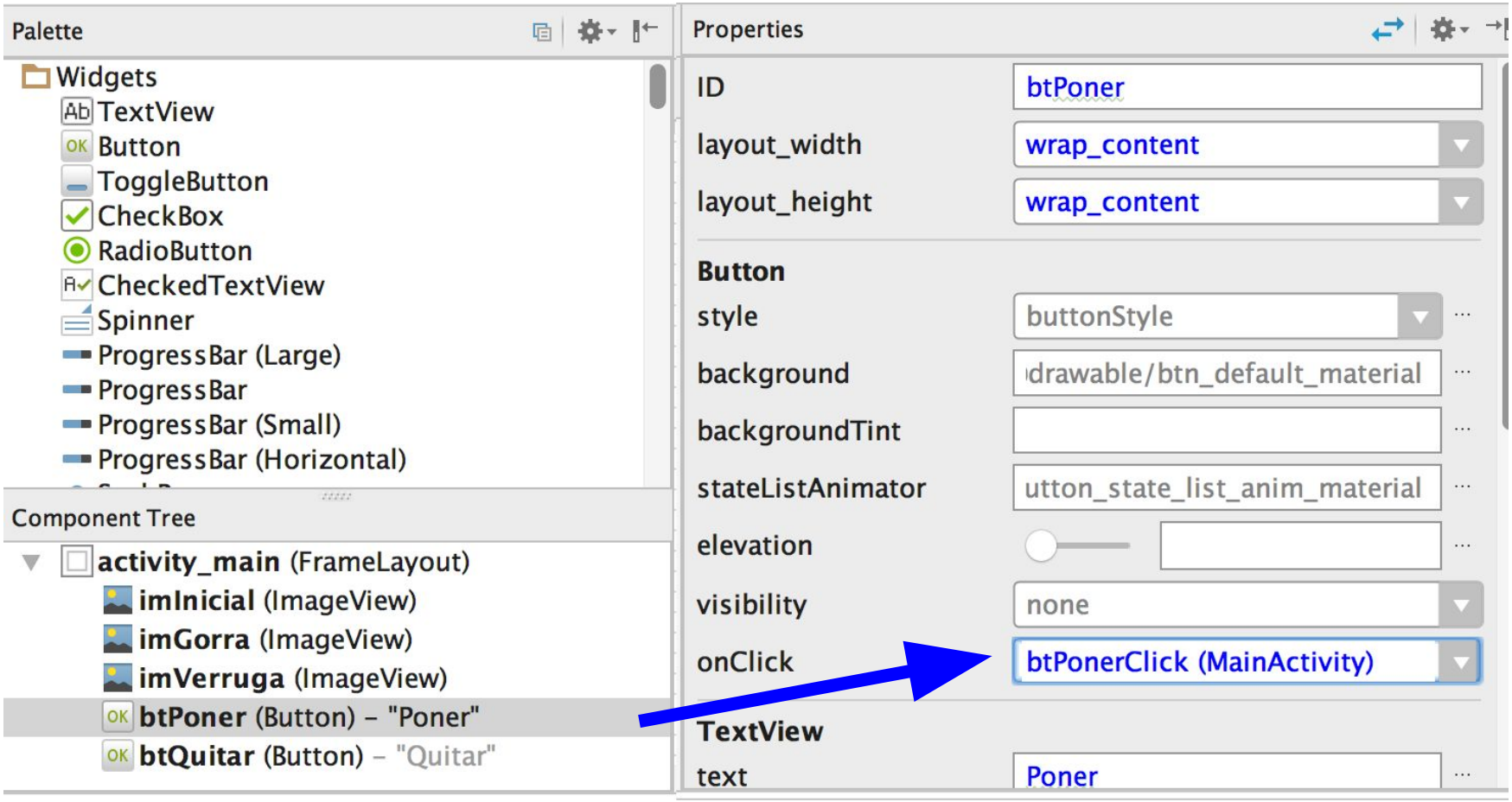
El Evento onClick

- Puede ser asignado directamente desde el propio fichero XML de la actividad

```
<Button
   android:text="Poner"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:id="@+id/btPoner"
   android:layout_marginTop="400dp"
   android:layout_marginLeft="40dp"
   android:onClick="btPonerClick" />
```



OnClick: en XML



EJERCICIO

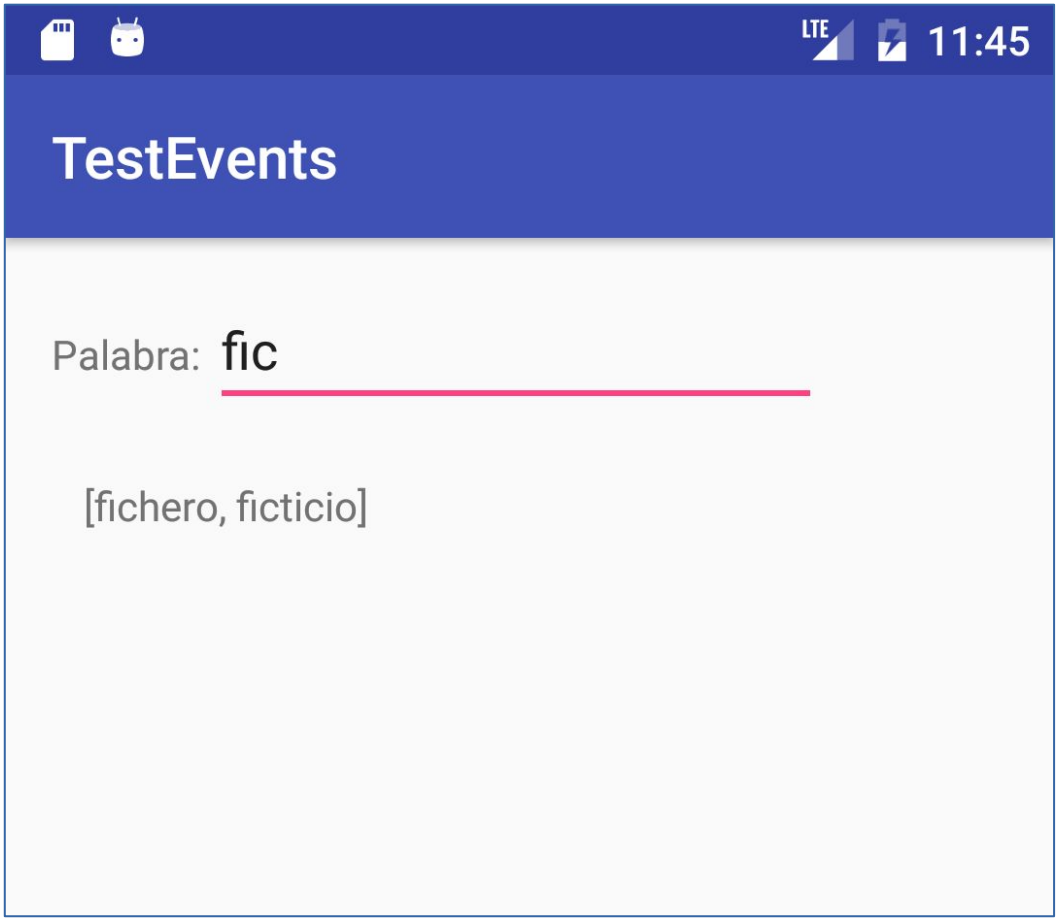
Documenta un par de eventos relativos a View

- Elige un par de eventos de la lista anterior o de la web de Android y escribe en un documento de texto lo siguiente:
  - Indica cuándo se dispara
  - Indica a qué widgets (o elemento de la app) afecta
  - Indica qué método de dicho widget permite añadirle/asignarle el *listener*
  - Busca un ejemplo de su uso y descríbelo
- No olvides subir tu documento en PDF a la plataforma.

Programación de la captura de eventos

- Necesitaremos:
  1. Crear el *callback* correspondiente
  2. "Enlazarlo" con el *widget* que lo puede capturar
- Distintas formas de programarlo:
  1. La *activity* implementa la interfaz
  2. El *widget* implementa la interfaz
  3. Se deriva una nueva interfaz

Ejemplo de TextWatcher



La activity implementa la interfaz

```
class MainActivity extends AppCompatActivity implements TextWatcher {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        etLetras=(EditText) findViewById(R.id.etPalabra);
        etLetras.addTextChangedListener(this);
    }
    @Override
    public void afterTextChanged(Editable campo) {
        if( flagEditText ) {
                    flagEditText=false;
            return;
        }
        tvCoincidencias=(TextView) findViewById(R.id.tvCoincidencias);
        ArrayList<String> coinciden=buscaCoincidencias( campo.toString() );
        tvCoincidencias.setText(coinciden.toString());
        if( coinciden.size()==1 ) {
            flagEditText=true;
            etLetras=(EditText) findViewById(R.id.etPalabra);
            etLetras.setText (coinciden.get(0).toString());
            etLetras.setSelection(etLetras.getText().length());
        }
    }
    @Override
    public void onTextChanged( CharSequence s, int start,  int before, int count) {}
    @Override
    public void beforeTextChanged( CharSequence s, int start,  int before, int count) {}
```

El widget implementa la interfaz

```
class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        etLetras=(EditText) findViewById(R.id.etPalabra);
        etLetras.addTextChangedListener(
          new TextWatcher() {
            @Override
            public void afterTextChanged(Editable campo) {
             if( flagEditText ) {
                  flagEditText=false;
                  return;
              }
              tvCoincidencias=(TextView) findViewById(R.id.tvCoincidencias);
              ArrayList<String> coinciden=buscaCoincidencias(campo.toString());
              tvCoincidencias.setText(coinciden.toString());
              if( coinciden.size()==1 ) {
                 flagEditText=true;
                 etLetras=(EditText) findViewById(R.id.etPalabra);
                 etLetras.setText (coinciden.get(0).toString());
                 etLetras.setSelection(etLetras.getText().length());
              } // fin del método afterTextChanged
```

El widget implementa la interfaz (II)

```
    @Override
    public void onTextChanged( CharSequence s, int start,  int before, int count) {}
    @Override
    public void beforeTextChanged( CharSequence s, int start,  int before, int count) {}
```

```
    private ArrayList<String> buscaCoincidencias( String inicio ) {
        ArrayList<String> toRet=new ArrayList<String>();
        for( int i=0; i<palabras.size(); ++i ) {
            if( palabras.get(i).toUpperCase().startsWith(inicio.toUpperCase()) ) toRet.add( palabras.get(i) );
        }
       return toRet;
     }
    }
  ); // Fin de la llamada al método addTextChangedListener
} // Fin del método onCreate de la activity
```

Una nueva clase implementa la interfaz

```
class MainActivity extends AppCompatActivity {

    // Clase para manejar los eventos
    class ManejadorEventos implements TextWatcher {
        @Override
        public void afterTextChanged(Editable campo) {
            if( flagEditText ) {
                flagEditText=false;
                return;
            }
            tvCoincidencias=(TextView) findViewById(R.id.tvCoincidencias);
            ArrayList<String> coinciden=buscaCoincidencias( campo.toString() );
            tvCoincidencias.setText(coinciden.toString())
            if( coinciden.size()==1 ) {
                flagEditText=true;
                etLetras=(EditText) findViewById(R.id.etPalabra);
                etLetras.setText (coinciden.get(0).toString());
                etLetras.setSelection(etLetras.getText().length());
            }
        } // Fin del método afterTextChanged
```

Una nueva clase implementa la interfaz (II)

```
@Override
public void onTextChanged( CharSequence s, int start,  int before, int count) {}
@Override
public void beforeTextChanged( CharSequence s, int start,  int before, int count) {}

private ArrayList<String> buscaCoincidencias( String inicio ) {
   ArrayList<String> toRet=new ArrayList<String>();
   for( int i=0; i<palabras.size(); ++i ) {
     if( palabras.get(i).toUpperCase().startsWith(inicio.toUpperCase()) )
                toRet.add( palabras.get(i) );
     }
     return toRet;
   }
} // Fin clase ManejadorEventos
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    etLetras=(EditText) findViewById(R.id.etPalabra);
    etLetras.addTextChangedListener( new ManejadorEventos() );
} // Fin del método onCreate
```

OnClick: en Java

```
public class MainActivity extends AppCompatActivity implements View.OnClickListener {
    Button btQuitar;
    int numCapas=1  ;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
         setContentView(R.layout.activity_main);
        btQuitar = (Button) findViewById(R.id.btQuitar);
        btQuitar.setOnClickListener(this);
    }
    @Override
    public void onClick(View bt) {
        ImageView tmpImg=null;
        if( numCapas<=0 ) return;
        switch (numCapas) {
            case 1: tmpImg=(ImageView) findViewById(R.id.imInicial); break;
            case 2: tmpImg=(ImageView) findViewById(R.id.imGorra); break;
            case 3: tmpImg=(ImageView) findViewById(R.id.imVerruga); break;
        }
        --numCapas;
        tmpImg.setVisibility(View.INVISIBLE);
    }
}
```

Problema si la propia actividad implementa la interfaz

- Aunque este es el método preferido por muchos desarrolladores…
- ...¿qué ocurre si varios *widgets* deben responder al mismo tipo de evento?
- Respuesta: Debemos desambiguar atendiendo al ID del *widget* que recibió el evento.

Desambiguación por ID

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

    Button btQuitar;
    Button btPoner;
    int numCapas=1  ;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
         setContentView(R.layout.activity_main);
        btQuitar = (Button) findViewById(R.id.btQuitar);
        btPoner = (Button) findViewById(R.id.btPoner);

        btQuitar.setOnClickListener(this);
        btPoner.setOnClickListener(this);
    }

    @Override
    public void onClick(View bt) {
        switch (bt.getId()) {
            case R.id.btQuitar: btQuitarClick(bt); break;
            case R.id.btPoner: btPonerClick(bt); break;
        }
    }
```

EJERCICIO

Importe de un pedido

MTV: Programar *callbacks* para eventos

- Realizar la app *“e-Food”* que permita pedir entre 0 y 10 unidades de 3 productos que aparecerán en pantalla relacionados con un restaurante de comida a domicilio.
- Al pulsar un botón “Actualizar” se debe presentar el importe total del importe pedido.
  - Cada producto tendrá un precio unitario fijo, el que tú desees.
  - Si al pulsar el botón, el número de unidades en algún producto fuese menor que 0, se cambiará a 0; si fuese mayor que 10, se cambiará a 10.
  - Puedes usar los *widgets* que desees.
  - Haz dos versiones usando dos de las tres formas que hemos visto de gestionar los eventos.
- Sube a la plataforma un documento PDF con una captura de pantalla de la aplicación y el código de los *callback* creados para gestionar los eventos.