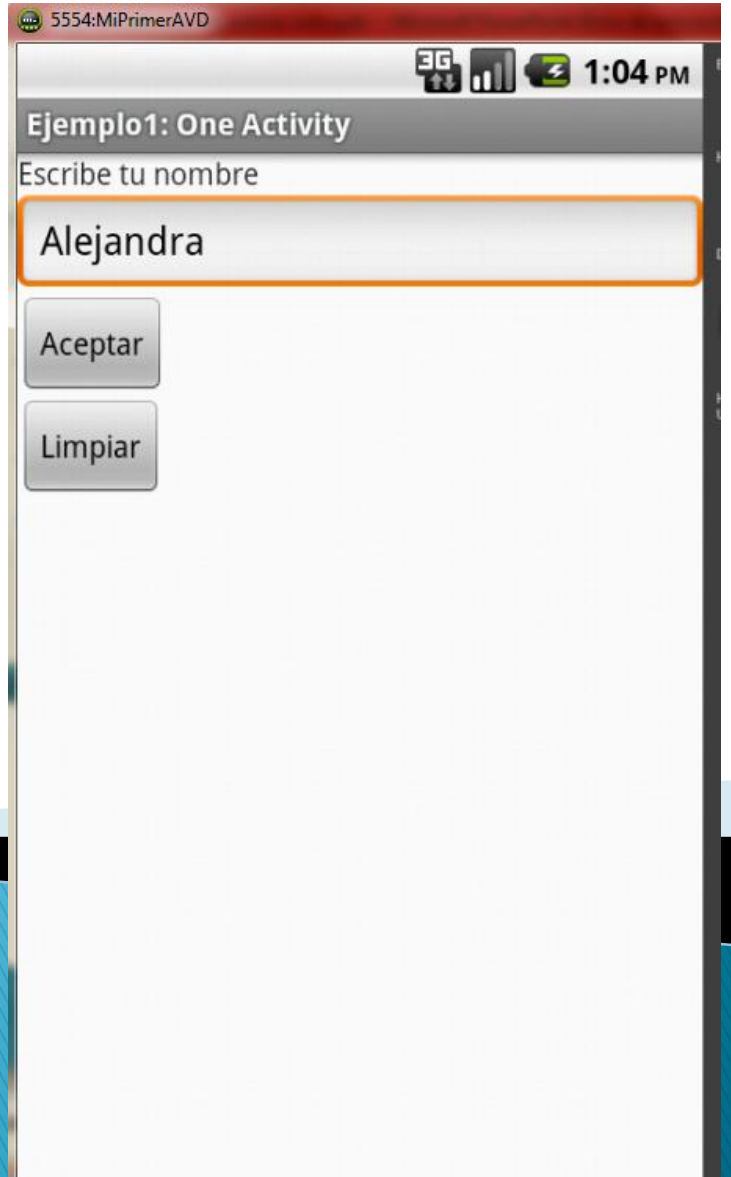
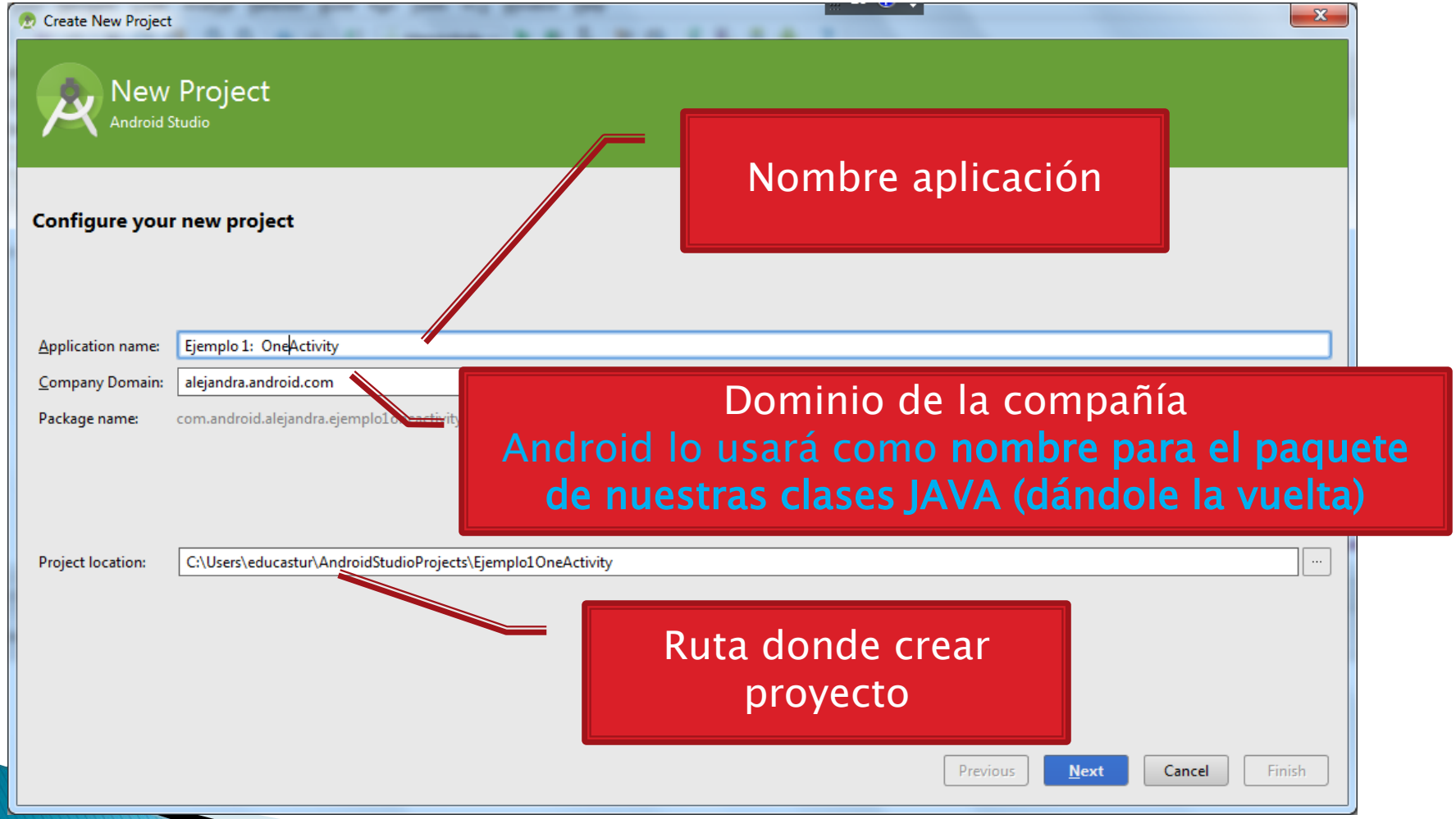
Ejemplo con una sola activity:

"Aplicación que pide que escribamos nuestro nombre"



- Vamos a crear un nuevo proyecto, como hicimos anteriormente: **FILE–NEW PROJECT:**



- Con esto tendremos simplemente un “**Hola Mundo**” como el que ya habíamos creado anteriormente.

- Vamos a **modificarlo** para conseguir la nueva app.

Modificaremos la pantalla principal que Android Studio creó automáticamente:

En Android, el **diseño** y la **lógica** de una **pantalla** están separados en dos ficheros distintos.

- **DISEÑO**: en el **fichero res/layout/activity_main.xml** (*diseño puramente visual de la pantalla definido como fichero XML*)
- **LÓGICA**: en el **fichero java/paquete.java/MainActivity.java,** (*código java que determina la lógica de la pantalla*).

PASO 1º) Modificaremos el **diseño**, añadiendo los **controles** (***views***) que nos hacen falta.

- Debéis sustituir el contenido del fichero **res/layout/activity_main.xml** por éste:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/LblNombre"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/nombre" />

    <EditText
        android:id="@+id/TxtNombre"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:inputType="text">
    </EditText>
```

Os dará un error el código aquí, porque aún no existen estas cadenas.

Luego las creamos

```
    <Button
        android:id="@+id/BtnAceptar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/aceptar" />

    <Button
        android:id="@+id/BtnLimpiar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/limpiar" />

</LinearLayout>
```

Explicación del código anterior: <LinearLayout…>

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >
    ……
</LinearLayout>
```

- Los **layout** son elementos no visibles que determinan cómo se van a distribuir en el espacio los controles que incluyamos en su interior.

- **LinearLayout:**
  - distribuye los controles simplemente uno tras otro y ,
  - en la orientación que indique su propiedad "android : orientation", que en este caso será "vertical".

Explicación del código anterior: <TextView…>, <EditText…>, <Button…>

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    ……
 <TextView
        android:id="@+id/LblNombre"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
       …… />
 <EditText
        android:id="@+id/TxtNombre"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        ……>
 </EditText>
 <Button
        android:id="@+id/BtnAceptar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content“
        …… />
 <Button
        …… />
</LinearLayout>
```

- Dentro del “<**LinearLayout**>” incluimos 4 **controles**:
  - Una etiqueta <TextView>
  - Un cuadro de texto <EditText>
  - Dos botones <Button>“ (*Aceptar y Limpiar*).

- Propiedades comunes de todos los controles:
  - android:id → ID del control. “@+id/…”
  - android:layout_height y
  - android:layout_width. Dimensiones del control con respecto al layout.

Mª Alejandra Tomás Fernández

Explicación del código anterior: <TextView…>, <Button…>

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/LinearLayout1"
    ……
    <TextView
        android:id="@+id/LblNombre"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/nombre" />
    ……
</LinearLayout>
```

- **ID del control**, permite identificarlo más tarde en nuestro código.
- El identificador se escribe precedido de "**@+id/**".
- Así, al compilarse el proyecto se genera automáticamente una nueva constante en la clase R para dicho control.
- ***Ejemplo***: como al cuadro de texto le hemos asignado el ID LblNombre, podremos más tarde acceder al él desde nuestro código haciendo referencia a la constante R.id.LblNombre.

- **Dimensiones del control** con respecto al layout que lo contiene.
- "**wrap_content**" indica que las dimensiones del control se ajustarán al contenido del mismo

- **Texto de la etiqueta:** hay dos formas de escribirlo:
  1. Escribiendo el **texto directamente**
  2. Usando **recursos "strings"** y aquí poniendo el nombre del string.

En la siguiente diapositiva lo veremos mejor

Explicación del código anterior:
android:text="@string/nombre"

- **android:text** indica el texto que aparece en el control.
- *Dos alternativas* de hacer esto:

1. **Directamente** como valor de la propiedad android:text.

```
<TextView
        android:id="@+id/LblNombre"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Escribe tu nombre: " />
```

2. Utilizar alguna de las **cadenas de texto** definidas en los **recursos** del proyecto (como ya vimos, en el fichero de recursos **res/values/strings.xml**, por ejemplo con identificador "*nombre*" y valor "*Escribe tu nombre:*".

   - Se indica como valor de la propiedad **android:text** **su identificador precedido del prefijo "@string/".**

(*esta 2ª es como se hizo y es la mejor forma*).

Explicación del código anterior:

¿cómo crear las cadenas de texto en el fichero: res/values/strings.xml

- Hacer doble clic sobre el fichero **res/values/strings.xml**.
- Se abre el fichero en modo gráfico. Pasarse al **modo texto "xml".**
- **Añadir** las siguientes líneas **al fichero**:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">Ejemplo1: One Activity</string>
    <string name="action_settings">Settings</string>
    <string name="hello_world">Hello world!</string>
    <string name="nombre">Escribe tu nombre</string>
    <string name="aceptar">Aceptar</string>
    <string name="limpiar">Limpiar</string>

</resources>
```

Admite formato mediante códigos HTML

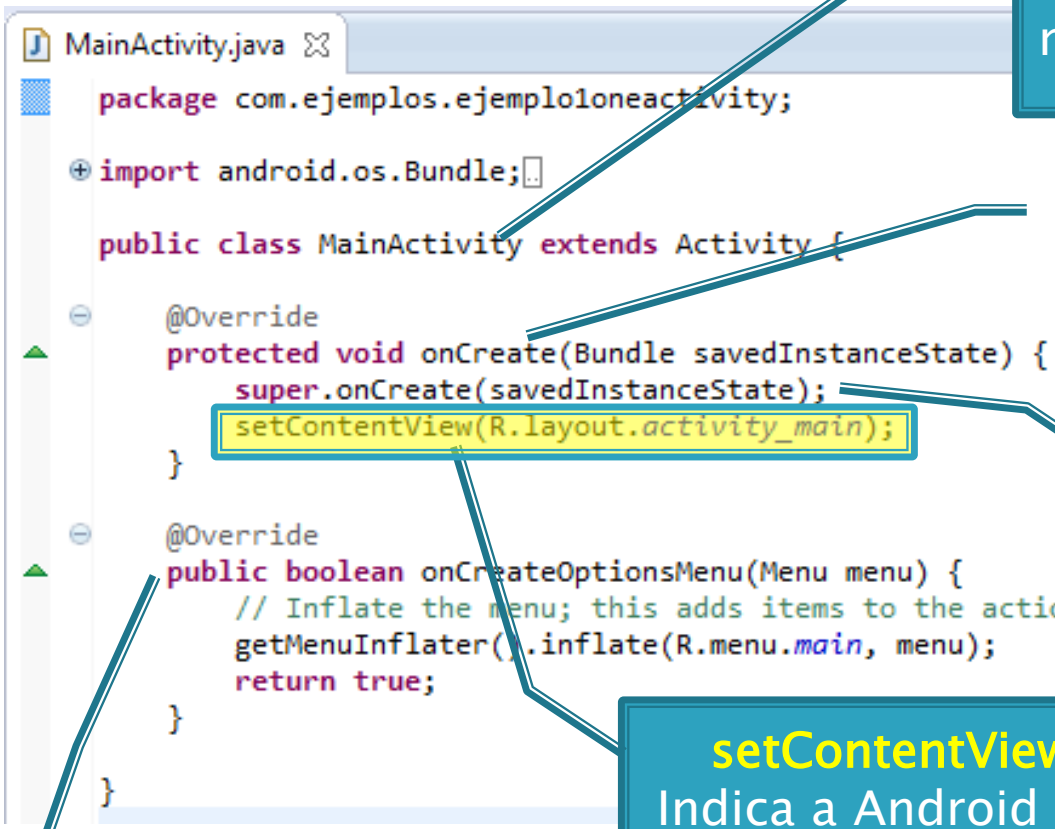
- Si **ejecutamos nuestra aplicación** veremos la pantalla deseada.
- Pero los **botones no** tienen **funcionalidad.**
- Con el **XML** sólo hemos hecho un **diseño gráfico.** Falta el código **JAVA** para darle **funcionalidad.**

PASO 2º) Añadiremos **funcionalidad** al botón "**Limpiar**" mediante código JAVA. (**lógica**).

Vamos a hacer que al pulsarlo, borre el texto que haya escrito en el cuadro de texto. Para ello:

- Abrimos el fichero **java/com.android.alejandra.ejemplo1oneactivity/MainActivity.java** haciendo doble-clic sobre él.

- Nos aparece esto:



Es la clase que representa nuestra pantalla (**ACTIVITY**)

Método que es **llamado por 1ª vez cuando se crea la Activity**

super.onCreate()
Llamada a su implementación en la clase padre

setContentView(R.layout.*activity_main*)
Indica a Android que debe **establecer** como **interfaz gráfica** de esta **activity** la definida en el recurso **R.layout.*activity_main***, que no es más que la que hemos especificado en el fichero **/res/layout/activity_main.xml**.

onCreateOptionsMenu()
Método que se utiliza para definir menús en la aplicación.
De momento no lo usaremos.

- Añadiremos ahora nuestro código:
  - **Todos los controles de la interfaz que se vayan a manipular** *(TextView, Button,…),* deben ser **declarados en Java**.
  - En nuestro caso el **EditText** y el **Button** *"Limpiar":*

```
EditText texto;
Button botonLimpiar;
```

  - Necesitamos **obtener una referencia a los controles** de la interfaz que hemos declarado.
  - Para ello usamos el *método* **findViewById() indicando el ID de cada control**, definidos como siempre en la clase R.
  - Todo esto se hace dentro del *método* **onCreate()** de la clase **MainActivity**, justo a continuación de la llamada a setContentView() que ya vimos:

```
texto = (EditText) findViewById(R.id.TxtNombre);
botonLimpiar = (Button) findViewById(R.id.BtnLimpiar);
```

- Implementamos el **evento "clic" del botón "Limpiar"** *(dentro del método onCreate)*:

```
botonLimpiar.setOnClickListener(new OnClickListener()
{

    @Override
    public void onClick(View v) {
    // TODO Auto-generated method stub


    }
});
```

Aquí pondremos el código que queremos ejecute el botón:

```
texto.setText("");
```

- El código completo que os debe quedar es:

```
public class MainActivity extends Activity {

    EditText texto;
    Button botonLimpiar;

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

          //Obtenemos una referencia a los controles de la interfaz
          texto = (EditText) findViewById(R.id.TxtNombre);
          botonLimpiar = (Button) findViewById(R.id.BtnLimpiar);

          //Implementamos el evento "click" del botón
          botonLimpiar.setOnClickListener(new OnClickListener() {

                  @Override
                  public void onClick(View v) {

                            texto.setText("");

                  }
          });
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
          // Inflate the menu; this adds items to the action bar if it is present.
          getMenuInflater().inflate(R.menu.main, menu);
          return true;
    }
}
```