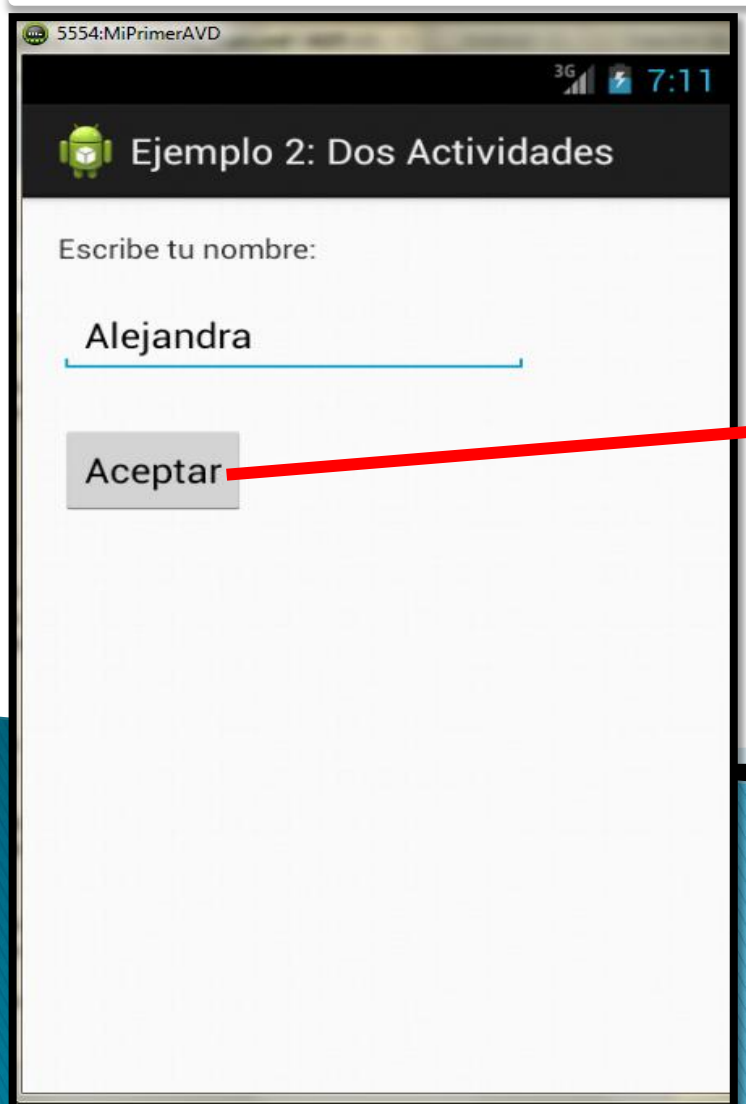
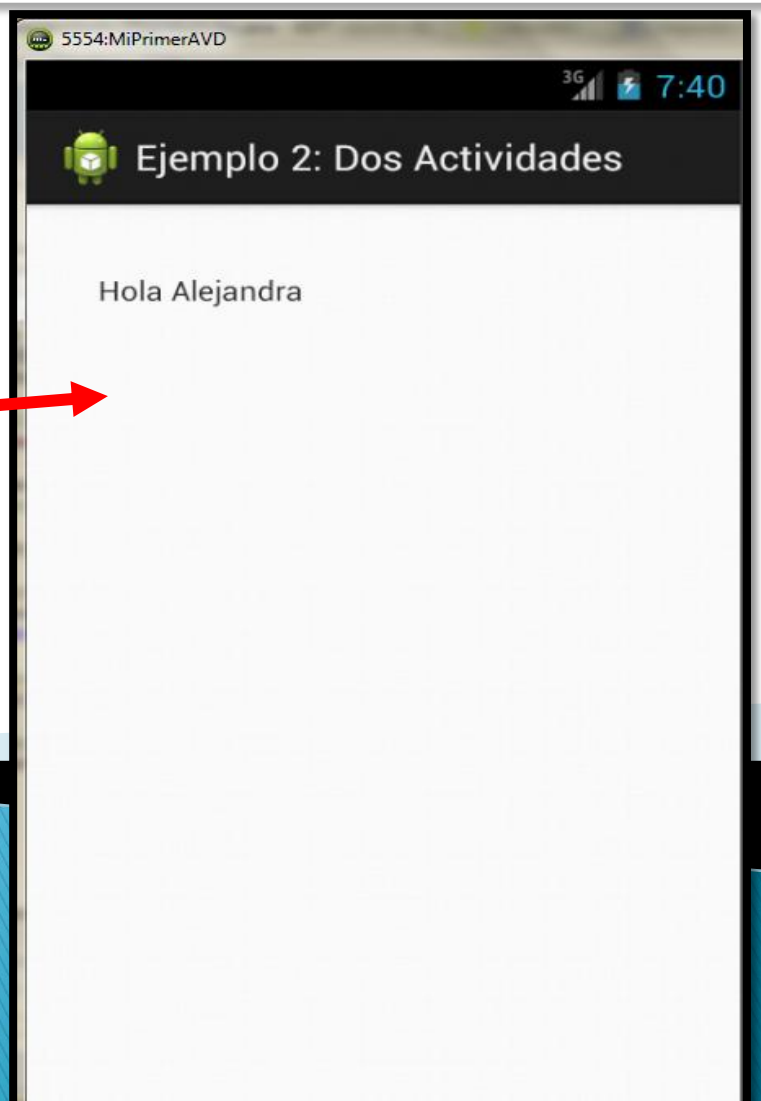
Ejemplo dos activities:

"Aplicación que pide que escribamos nuestro nombre y nos saluda"





- Vamos a crear un nuevo proyecto, como hicimos anteriormente: **FILE-NEW PROJECT** y le ponéis de nombre "**Ejemplo2DosActividades**".
- Con esto tendremos simplemente un "**Hola Mundo**" como el que ya habíamos creado anteriormente.
- Vamos a **modificarlo** para conseguir la nueva app.

- Como nuestra aplicación tiene 2 pantallas, **necesitaremos**:
  - 2 Activity.
- **Cada activity tendrá su propio diseño de interfaz de usuario**:
  - Un layout para cada activity (DISEÑO DE LA PANTALLA DE LA ACTIVITY). *Definido en el fichero* app/src/main/res/layout/xxx.xml
  - Una clase java para cada Activity (LOGICA DE LA PANTALLA). *Definida en el fichero* *app/s*rc/main/paquete.java/yyy.java

1º) Haremos el **diseño** de nuestra **pantalla principal**, modificando la que Android Studio nos ha creado por defecto:

1.1) Modificamos el aspecto de la ventana principal **añadiendo los controles** (***views***) que vemos en la 1ª diapositiva de esta presentación.

Para ello, vamos a **sustituir** el contenido del fichero *res/layout/activity_main.xml* por el siguiente:

```
<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>

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

</LinearLayout>
```

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

Luego las creamos

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>
```

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 3 **controles**:
  - Una etiqueta <TextView>
  - Un cuadro de texto <EditText>
  - Un botón <Button>“ (*Aceptar*).

- **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.

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: <EditText…>

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

 ……
</LinearLayout>
```

Tipo de contenido va a albergar el control, en este caso texto normal (valor "**text**"), aunque podría haber sido una contraseña (textPassword), un teléfono (phone), una fecha (date), ….

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**:

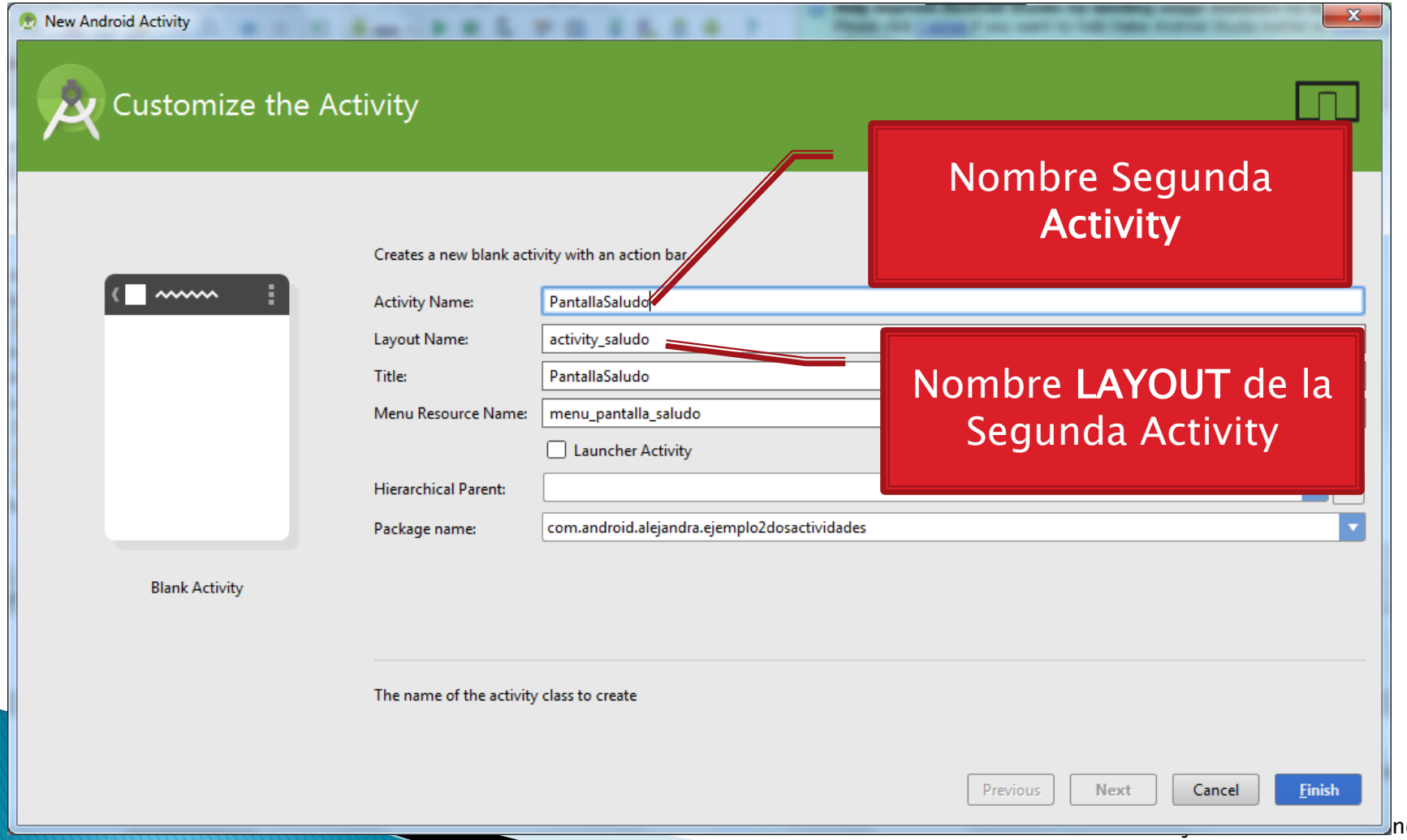
```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <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>

</resources>
```

2º) Haremos la segunda **pantalla, la que nos saluda**:

2.1) Para ello, **creamos una nueva Activity** pulsando **botón derecho** en la carpeta **src/main/java** y elegimos **New→Activity→Blank Activity**



2.2) Android Studio creará entonces el **archivo JAVA** de la nueva Activiy, su **archivo LAYOUT** y su **archivo MENÚ**.

2.3) Abrimos el nuevo layout (el de la 2ª Activity) y **accederemos a la solapa de código para modificar directamente el contenido XML del fichero**:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

   <TextView
        android:id="@+id/TxtSaludo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="" />

</LinearLayout>
```

- Si **ejecutamos nuestra aplicación** veremos la pantalla principal solamente.
- Además el **botón no** tienen **funcionalidad.**
- Con el **XML** sólo hemos hecho un **diseño gráfico.** Falta el código **JAVA** para darle **funcionalidad.**

3º) Añadiremos **funcionalidad** al botón "**Aceptar**" mediante código JAVA. (**lógica**):

Vamos a hacer que al pulsarlo, muestre una nueva pantalla con el texto "Hola" y el nombre que haya escrito en el cuadro de texto. Para ello:

3.1) **Vamos a la pantalla principal "MainActivity.java" a añadir el código necesario al botón "Aceptar":**

- **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** ***"Aceptar":***

```
final EditText txtNombre;
final Button btnAceptar;
```

- 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:

```
txtNombre= (EditText) findViewById(R.id.TxtNombre);
btnAceptar = (Button) findViewById(R.id.BtnAceptar);
```

3.2) Implementar las **acciones a tomar cuando pulsemos el botón** de la pantalla.

- Para ello, implementamos el **evento "clic" del botón "Limpiar"** *(dentro del método onCreate)*:

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

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

    }
});
```

Aquí pondremos el código que queremos ejecute el botón:

En nuestro caso, habrá que crear un "Intent" y llamar a la segunda activity con ese "Intent"

- El código completo que os debe quedar es:

```
public class MainActivity extends Activity {

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

        //declaramos y obtenemos una referencia a los controles de la interfaz
        final EditText txtNombre= (EditText) findViewById(R.id.TxtNombre);
        final Button btnAceptar= (Button) findViewById(R.id.BtnAceptar);

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

            @Override
            public void onClick(View v) {
                //Creamos el Intent
                Intent intent = new Intent(MainActivity.this, PantallaSaludo.class);

                //Creamos la información a pasar entre actividades
                Bundle b = new Bundle();
                b.putString("NOMBRE", txtNombre.getText().toString());

                //Añadimos la información al intent
                intent.putExtras(b);

                //Iniciamos la nueva actividad
                startActivity(intent);
            }

        });
    }
    ..........
}
```

4º) Con esto, ya **hemos finalizado** la "Activity" principal de la aplicación "**MainActivity**".

5º) Debemos **modificar la segunda "Activity" para que en su cuadro de texto, aparezca el saludo con el nombre que recibe de la primera Activity**:

Para ello:

5.1) **Ampliamos** el **método "onCreate"** de la clase "**PantallaSaludo**", obteniendo las *referencias a los objetos* que manipularemos, que será sólo la *etiqueta de texto:*

```
TextView txtSaludo = (TextView) findViewById(R.id.TxtSaludo);
```

5.2) **Recuperamos la información pasada desde la activity principal** y se la **asignamos como texto a la etiqueta**:

- **Accedemos al Intent** que dio lugar a la actividad actual mediante el método "**getIntent()**". *Este método devuelve un "Intent".*

- **Recuperamos** la **información asociada al intent** obtenido (*objeto Bundle*) mediante el **método "getExtras()".**

Estos dos pasos: acceder al intent y extraer su Bundle asociado, los hacemos en uno solo:

```
Bundle bundle = this.getIntent().getExtras();
```

- Construimos el texto de la etiqueta mediante su método "setText(texto)" y recuperando el valor de nuestra clave almacenada en el objeto Bundle mediante "getString(clave)":

```
txtSaludo.setText("Hola " + bundle.getString("NOMBRE"));
```

El código completo de la clase "PantallaSaludo" es:

```
public class PantallaSaludo extends Activity {
 @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_saludo);

        //obtenemos las referencias a los objetos que manipularemos
        TextView txtSaludo = (TextView) findViewById(R.id.TxtSaludo);

        //Recuperamos la información pasada en el intent
        Bundle bundle = this.getIntent().getExtras();

        //Construimos el mensaje a mostrar
        txtSaludo.setText("Hola " + bundle.getString("NOMBRE"));


    }
}
```

6º) SOLO NOS QUEDA COMPROBAR QUE NUESTROS COMPONENTES ESTÁN DECLARADOS EN EL "AndroidManifest.xml":

6.1. La **actividad principal** ya aparece **por defecto declarada**.

6.2. **Comprobamos** sólo la **segunda activity: "PantallaSaludo"**:

```
<activity android:name=".PantallaSaludo"
          android:label="@string/title_activity_saludo" >

</activity>
```

Este "string" hay que definirlo (si no está ya) en **res/values/strings.xml** añadiendo la línea:

```
<string name="title_activity_saludo">Hola amig@</string>
```

YA PODÉIS EJECUTARLO Y PROBARLO!