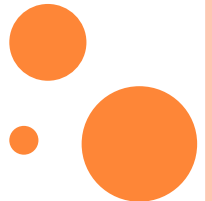
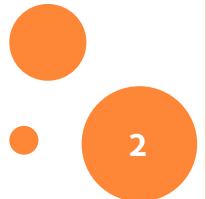


CONTEXT E INTENTS



CONTEXT E INTENTS

- El contexto (clase *Context*):
 - Da acceso a la información global de una aplicación.
 - Por medio del contexto, podemos:
 - Acceder a las clases pertenecientes a la *app*.
 - Comunicarnos con otros componentes.
 - Da acceso a los *Servicios del Sistema*:
 - AlarmManager
 - AudioManager
 - LocationManager
 - ...
 - La clase *Activity* es una subclase de la clase *Context*.

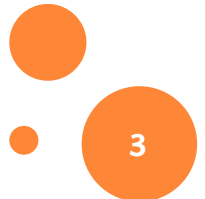


CONTEXT E INTENTS

○ El contexto

- Métodos útiles asociados al contexto:

- *getPackageName()*: devuelve el nombre del paquete de la app.
- *databaseList()*: devuelve los nombres de las bases de datos que hemos creado para esa app.
- *fileList()*: devuelve los nombres de los ficheros creados para la app.
- *getCacheDir()*: devuelve el path absoluto al directorio caché de la app.
- *getContentResolver()*: devuelve una instancia de un *Content Resolver* para nuestra app.
- *getFileDir()*: devuelve el path absoluto al directorio donde se crean los ficheros de nuestra app.
- *getSharedPreferences()*: devuelve los contenidos de un fichero de preferencias.



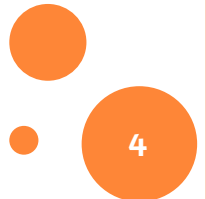
CONTEXT E INTENTS

○ El contexto

• Ejemplo: la clase *View*

- Se basa en el contexto, para construir el objeto *View*.
- Necesita saber el tema de la *app* y otro tipo de información.
- Cuando se construye un *View*, se le puede pasar un contexto diferente, de forma que tengamos acceso a recursos de otras actividades.
- Para saber en qué contexto se está ejecutando un *view*, podemos *llamar al método*:

```
getContext();
```

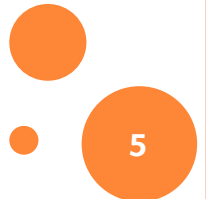


CONTEXT E INTENTS

- El contexto

- Tres tipos de contexto:

- Application context: unido al ciclo de vida de la aplicación. Es único para la aplicación y engloba el resto de contextos
 - Activity context: unido al ciclo de vida de la actividad
 - Service context: unido al ciclo de vida de un servicio

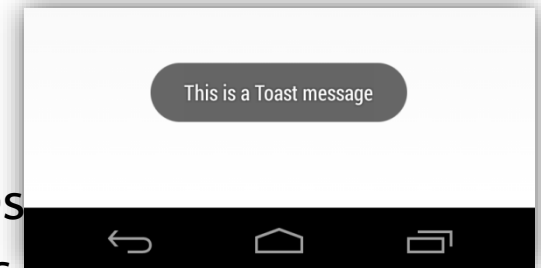


CONTEXT E INTENTS

- El contexto

- Application context:

- Se obtiene mediante: `getApplicationContext();`
 - También se puede usar `getApplication()`, porque la clase Application extiende la clase Context.
 - El Application context es el mismo durante el ciclo de vida de la app.
 - Ejemplos de uso:
 - Para mostrar un mensaje Toast.
 - Para acceder a recursos y configuraciones compartidas entre diferentes actividades.



CONTEXT E INTENTS

○ El contexto

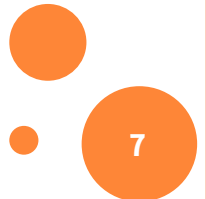
- Activity context:

- Se une al ciclo de vida de la actividad, y se destruye con la actividad.
- La clase Activity extiende la clase Context ⇒

Una actividad es un Context

- Por tanto, como el contexto de una actividad se puede usar el mismo nombre de la clase de la actividad, o la palabra reservada "this".
- Para obtener el contexto de una actividad:

```
getBaseContext();
```



CONTEXT E INTENTS

○ El contexto

- ¿Qué contexto utilizar?
 - Como norma general, el más cercano
 - Hay que tener en cuenta el ciclo de vida de los elementos que asociamos al context
 - Si debe “vivir” más allá de una actividad (variables tipo static, clases singleton, etc.)
 - Usaremos el contexto de la aplicación

Un mal uso del contexto puede generar que la aplicación consuma mucha memoria porque nunca se libere (memory leaks)

También se producen memory leaks a través del uso de clases anidadadas

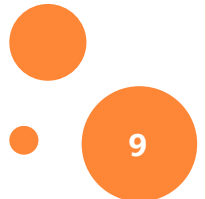
Si la clase anidada se referencia desde fuera de la actividad: static



CONTEXT E INTENTS

○ Intents

- Son objetos que se usan para activar una acción. Es la forma de llamada y comunicación entre actividades
- Tres casos típicos:
 - Comenzar una actividad
 - Comenzar un servicio
 - Enviar un mensaje de broadcast



CONTEXT E INTENTS

○ Tipos de *intents*

- **Intents explícitos**: se indica el **nombre de la clase** del componente que se quiere activar.
Son los que se utilizan entre las actividades pertenecientes a una misma app.
- **Intents implícitos**: no se indica el nombre de la clase del componente que se quiere activar, sino **una acción** general.
- **Generalmente, se abrirá otra app**. P. ej.: para mostrar la localización del usuario en un mapa, se puede utilizar un *intent* implícito, para pedir que se abra alguna de las *apps* que sean capaces de ello.



CONTEXT E INTENTS

○ Tipos de *intents*

- Explícitos: se indica qué actividad se quiere abrir

```
Intent i = new Intent (this, SegundaActividad.class);  
startActivity(i);
```

- Implícitos: se deja que sea el S.O. el que decida qué actividad ejecutar para ese intent

```
Intent i = new Intent(Intent.ACTION_VIEW, Uri.parse("http://www.ehu.eus"));  
startActivity(i);
```

Si hay definida más de una aplicación posible, se le pregunta al usuario



CONTEXT E INTENTS

○ Intents explícitos

- Permiten el paso de información de una actividad a otra

```
Intent i = new Intent (this, SegundaActividad.class);  
i.putExtra("nombre1",valor);  
i.putExtra("nombre2", valor2);  
startActivity(i);
```

En la actividad que
llama

En la actividad
llamada

```
Bundle extras = getIntent().getExtras();  
if (extras != null) {  
    String valor= extras.getString("nombre1");  
    int valor2= extras.getInt("nombre2");  
}
```



CONTEXT E INTENTS

○ Intents explícitos

- Permiten recoger resultados de una en otra.
- Desde el paquete AndroidX: *startActivityForResult*
Deprecated

```
Intent intent= new Intent(MainActivity.this,SegundaActividad.class);  
startActivityForResult(intent, 666);
```

.....

Se ejecuta al volver
de la otra actividad

Se definen códigos para distinguir
distintas llamadas

```
protected void onActivityResult(int requestCode, int resultCode, Intent data)  
{  
    super.onActivityResult(requestCode, resultCode, data);  
    // TRATAR EL RESULTADO:  
    if (requestCode == 666 && resultCode == RESULT_OK) {  
        String requiredValue = data.getStringExtra("variable_01");  
    }  
}
```



CONTEXT E INTENTS

○ Intents explícitos

- En AndroidX: permiten recoger resultados de una en otra.

```
ActivityResultLauncher<Intent> startActivityIntent =  
registerForActivityResult(new  
ActivityResultContracts.StartActivityForResult(),  
    new ActivityResultCallback<ActivityResult>() {  
        @Override  
        void onActivityResult(ActivityResult result) {  
            // Add same code that you want to add in onActivityResult  
            //method  
            if (result.getResultCode() == RESULT_OK) {  
                requiredValue =  
                    result.getData().getStringExtra("variable_01");  
            }  
        }  
    });
```

Recoge el resultado al volver

Creamos el objeto para recoger resultado de otra actividad

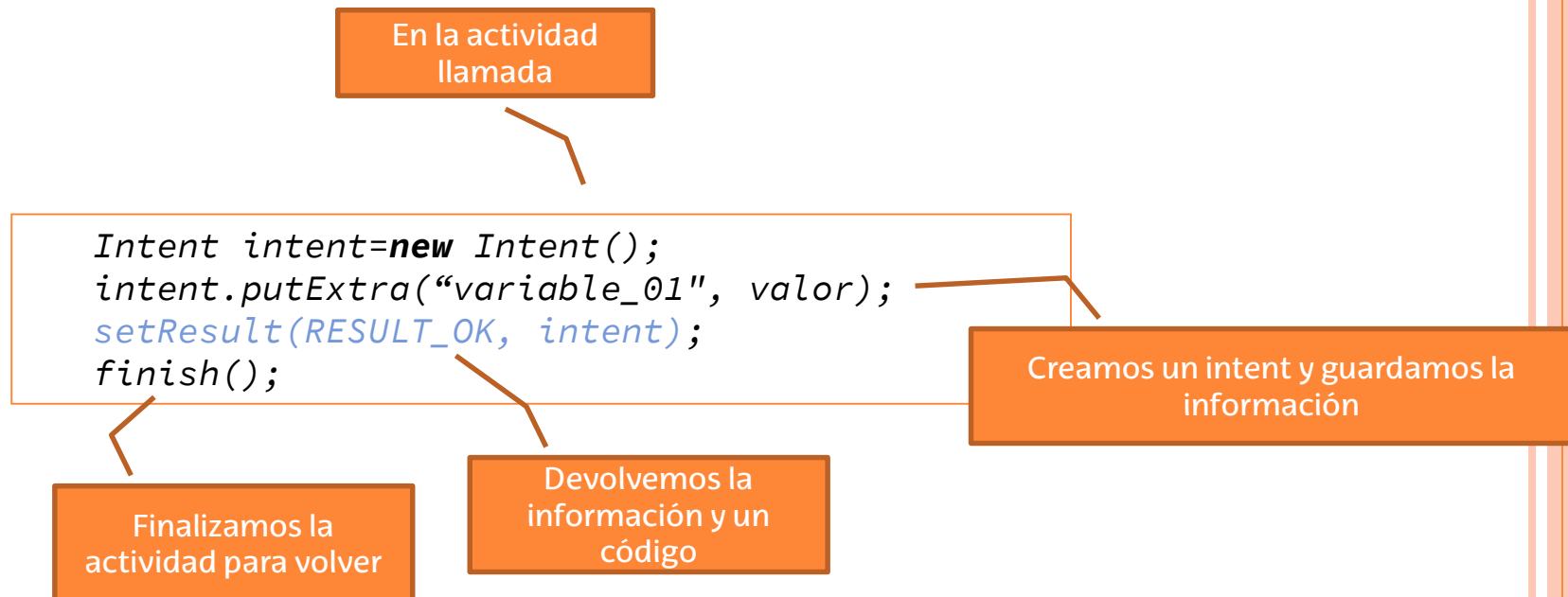
```
Intent intent = new Intent(MainActivity.this, MainActivity2.class);  
startActivityIntent.launch(intent);
```

Lanza la segunda actividad



CONTEXT E INTENTS

- Intents explícitos
 - Permiten recoger resultados de una en otra



CONTEXT E INTENTS

○ Intents explícitos

- Para poder llamar a una actividad mediante un *intent* debe estar definida en el *ManifestFile.xml*

```
<manifest xmlns:android=http://schemas.android.com/apk/res/android
.....
<activity
    android:name="com.example.Aplicacion.SegundaActividad"
    android:label="@string/title_activity_segunda_actividad">
</activity>
```

Si definimos la actividad desde las opciones de Android Studio, se añade al manifiesto automáticamente



CONTEXT E INTENTS

- **Ejercicio 1 (a):** Cread una actividad que tenga un TextView (con texto inicial "...") y un botón, que, al pulsarlo, llame a otra actividad.
 - Para crear un método que se ejecute al pulsar el botón, podemos usar el atributo "onClick" en el layout, y poner ahí el nombre del método que vamos a crear.
 - Hay que crear ese método en el código

Debe tener como parámetro una vista

```
public void EquivalenteOnClickListener(View v){  
  
}
```



CONTEXT E INTENTS

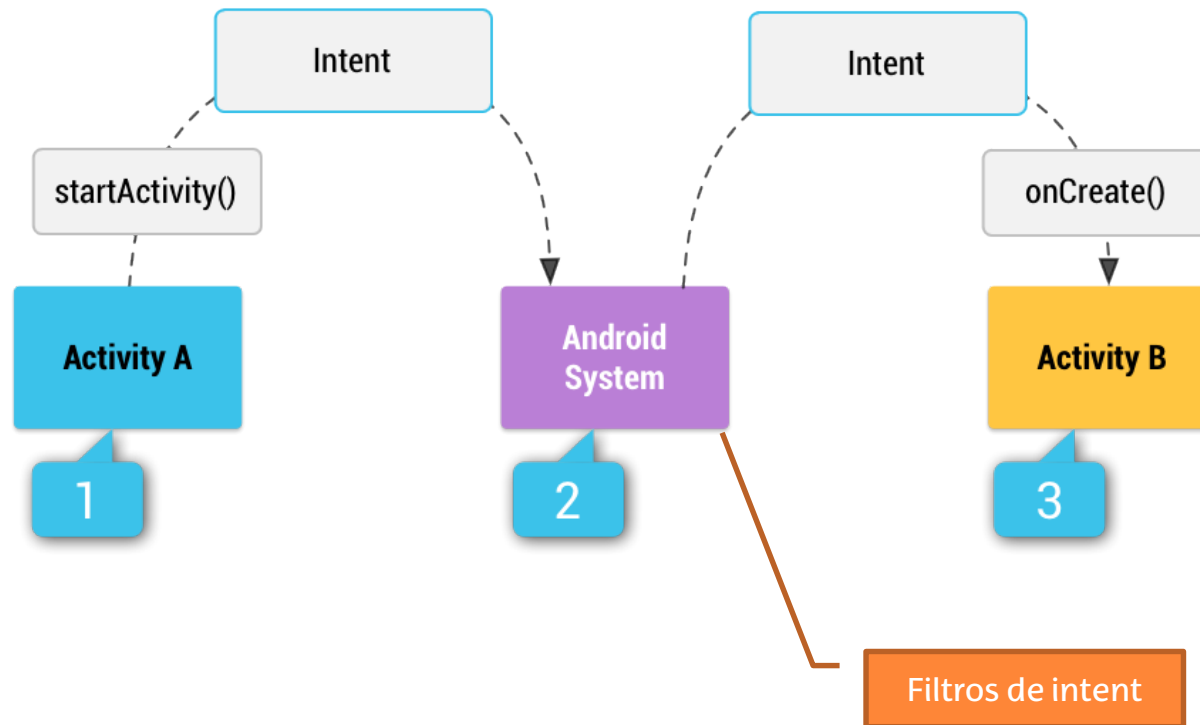
- **Ejercicio 1 (b):** En la segunda actividad, el usuario pondrá un texto cualquiera en un EditText, y, al pulsar en un botón, se devolverá el String del EditText a la primera actividad y se mostrará en el TextView



CONTEXT E INTENTS

○ Intents implícitos

- Es el sistema operativo el que decide qué actividad hay que lanzar en función de la acción a realizar



CONTEXT E INTENTS

◦ Intents implícitos

- Son *intents*
 - Que abren otras aplicaciones
 - Que acceden a datos en el dispositivo móvil

```
Intent i = new Intent(Intent.ACTION_CALL, Uri.parse("tel:946123456"));
startActivity(i);
```

• Acciones

- ACTION_CALL
- ACTION_DIAL
- ACTION_VIEW
- ACTION_TIME_CHANGED
- ACTION_PACKAGE_ADDED
-

Acción

Datos para la
acción

de actividad

de broadcast

Listado completo en:

<http://developer.android.com/reference/android/content/Intent.html>



CONTEXT E INTENTS

○ Intents implícitos

• Ejemplos

```
new Intent(Intent.ACTION_CALL, Uri.parse("tel:946123456"));  
new Intent(Intent.ACTION_DIAL, Uri.parse("tel:946123456"));
```

Llamar

Marcar

```
new Intent(Intent.ACTION_VIEW, Uri.parse("content://contacts/people"));  
new Intent(Intent.ACTION_VIEW, Uri.parse("http://www.ehu.es"));
```

Mostrar los
contactos

Mostrar en el
navegador

```
new Intent(Intent.ACTION_SENDTO, Uri.parse("mailto:mikel.v@ehu.eus"));
```

```
new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
```

Abrir redacción de mensaje
con destinatario

Abrir la aplicación de la cámara para
sacar una foto



CONTEXT E INTENTS

- Intents implícitos
 - Algunos necesitan permisos del usuario

```
new Intent(Intent.ACTION_CALL, Uri.parse("tel:946123456"));
```



```
<uses-permission android:name="android.permission.CALL_PHONE"/>
```

No es lo mismo acceder a la cámara (necesita permisos) que lanzar la aplicación que gestiona la cámara (no necesita permisos)



CONTEXT E INTENTS

- **Ejercicio 2:** Cread una actividad que tenga un EditText para introducir una dirección web y un botón que, al pulsarlo, abra el navegador para visitar dicha dirección.



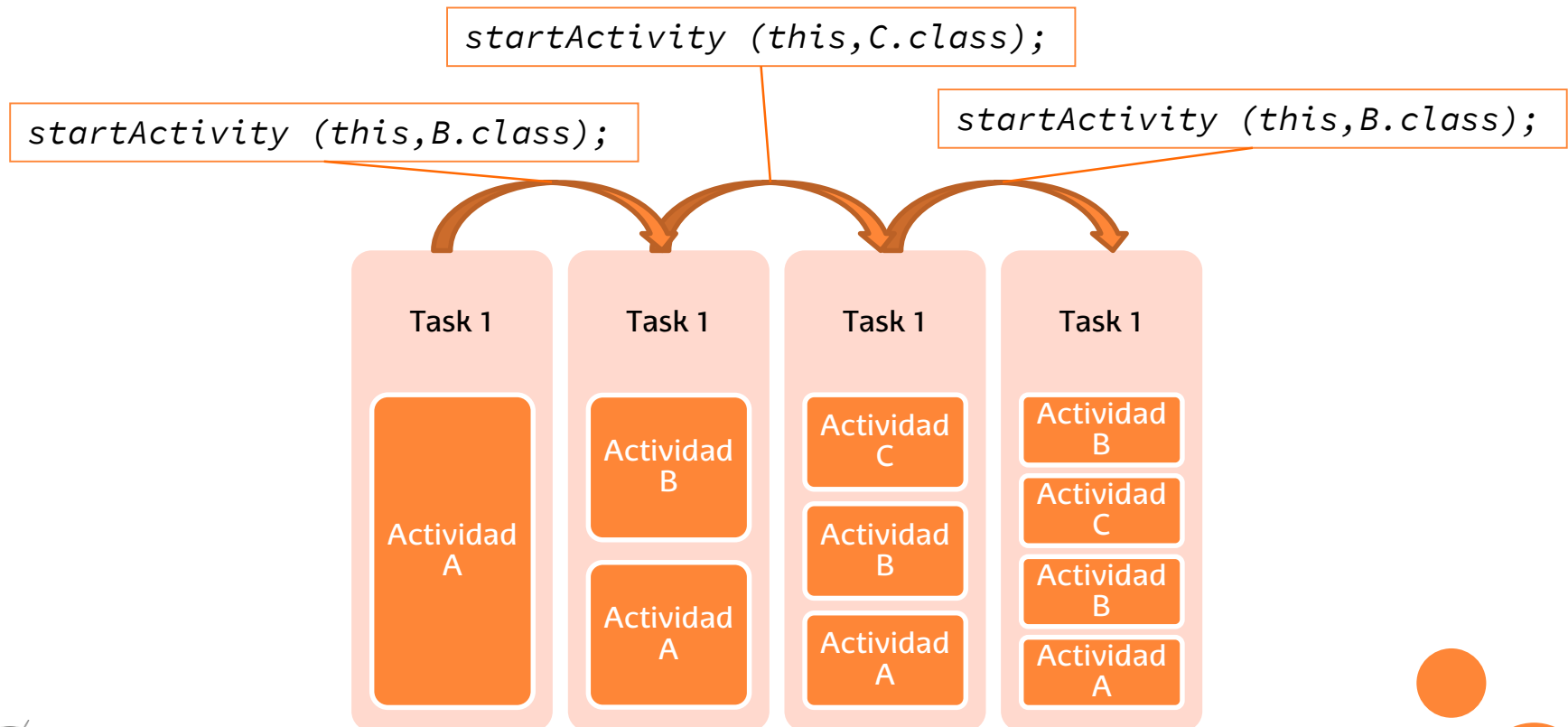
CONTEXT E INTENTS

- La pila de actividades
 - Gestionada mediante contenedores llamados tareas (*Tasks*)
 - Por defecto todas las actividades de una aplicación se lanzan en la misma tarea
 - En cada tarea hay una lista LIFO
 - Cada vez que se ejecuta *startActivity()* se genera una nueva instancia de la actividad en la pila
 - El botón "*back*" destruye la tarea en la cabecera y recupera la anterior
 - En el estado en el que estaba (los datos se mantienen)
 - Si la pila está vacía, se vuelve al escritorio
 - El estado se puede perder a consecuencia de la gestión de memoria



CONTEXT E INTENTS

- La pila de actividades
 - Comportamiento standard



CONTEXT E INTENTS

- La pila de actividades

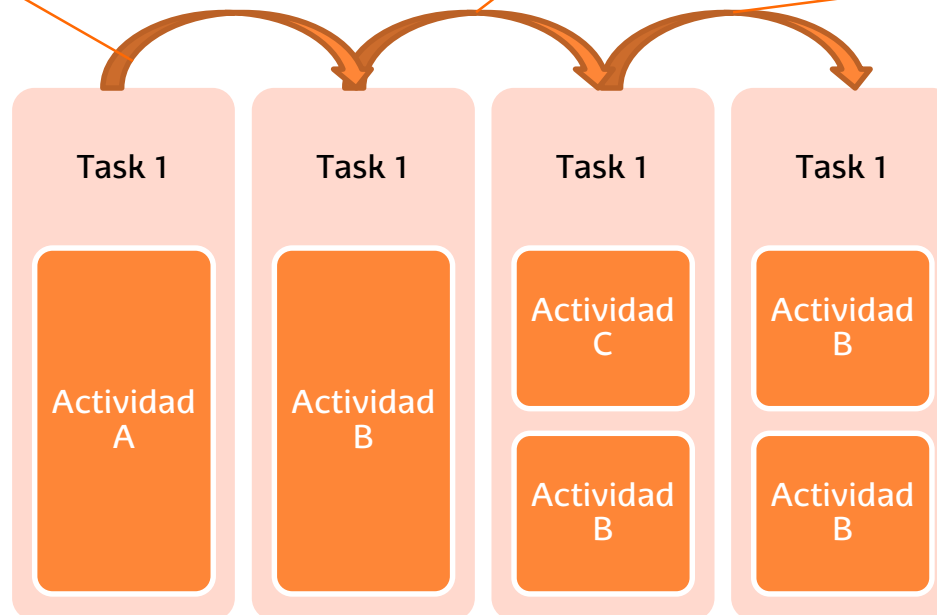
- Se puede modificar su comportamiento mediante la instrucción *finish()*

- Elimina esa actividad de la pila

```
startActivity (this,C.class);
```

```
startActivity (this,B.class);  
finish();
```

```
startActivity (this,B.class);  
finish();
```



CONTEXT E INTENTS

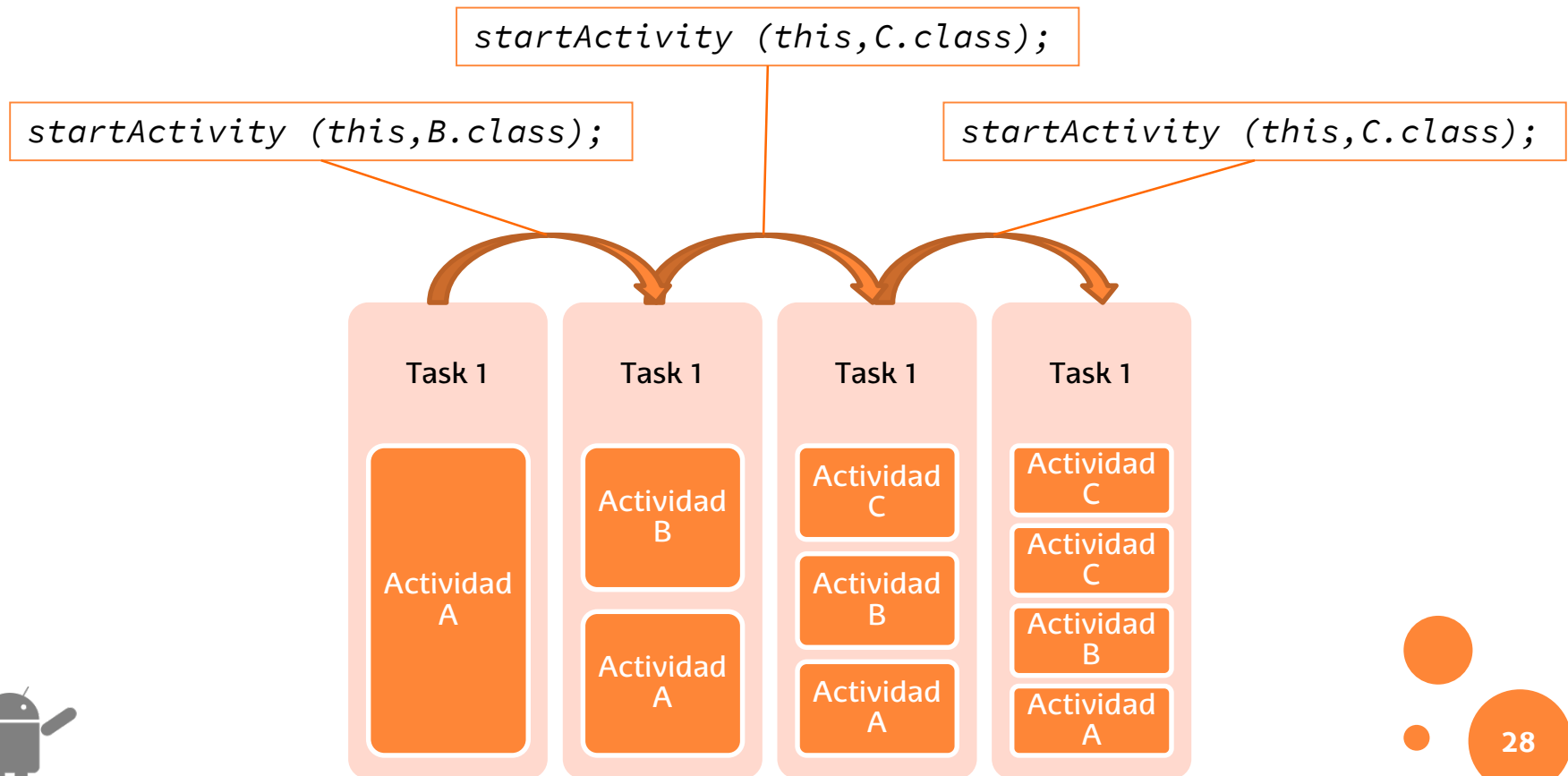
○ La pila de actividades

- También se puede usar el atributo *launchMode* de la actividad en el manifiesto
 - Define el comportamiento por defecto al llamar a esa actividad
 - android:launchMode="standard" → Es el comportamiento habitual, no se suele poner
 - android:launchMode="singleTop" → Si la actividad que se lanza está en la cabecera de la pila, no se crea una nueva instancia, se recupera la existente
 - Se lanza el método *onNewIntent()*
 - android:launchMode="singleTask" → Si la actividad que se lanza está en la pila, no se crea una nueva instancia, se recupera la existente
 - Se lanza el método *onNewIntent()*



CONTEXT E INTENTS

- La pila de actividades
 - android:launchMode="standard"



CONTEXT E INTENTS

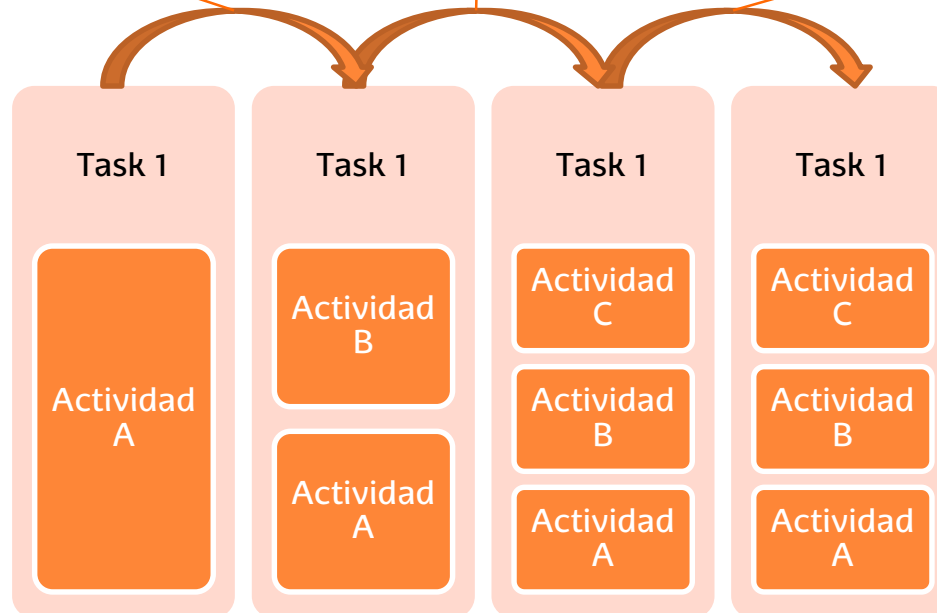
- La pila de actividades

- android:launchMode="singleTop" (en la actividad C)

```
startActivity (this,C.class);
```

```
startActivity (this,B.class);
```

```
startActivity (this,C.class);
```



CONTEXT E INTENTS

○ Diferencias entre los modos de lanzamiento

Casos de uso	Modo de lanzamiento	¿Varias instancias?	Comentarios
Lanzamientos convencionales para la mayoría de las actividades	"standard"	Sí	Predeterminado. El sistema siempre crea una nueva instancia de la actividad en la tarea destino y direcciona la intent hacia ella.
	"singleTop"	Condionalmente	Si una instancia de la actividad ya existe al comienzo de la tarea de destino, el sistema direcciona la intent hacia dicha instancia mediante un llamado a su método <code>onNewIntent()</code> , en lugar de crear una nueva instancia de la actividad.
Lanzamientos especiales (no se recomienda para usos generales)	"singleTask"	No	El sistema crea la actividad en la raíz de una tarea nueva y direcciona la intent hacia ella. Sin embargo, si ya existe una instancia de la actividad, el sistema direcciona la intent hacia la instancia existente mediante una llamada a su método <code>onNewIntent()</code> , en lugar de crear una nueva.
	"singleInstance"	No	Es igual que "singleTask", con la excepción de que el sistema no lanza otras actividades en la tarea que contiene la instancia. La actividad siempre es el único miembro de su tarea.



CONTEXT E INTENTS

○ La pila de actividades

- Se puede modificar el comportamiento de una llamada concreta a través de los FLAGS del Intent
 - Tienen prioridad sobre lo definido en el manifiesto
 - FLAG_ACTIVITY_SINGLE_TOP → Si la actividad está en la cabecera de la pila, no genera una nueva instancia (equivalente a "singleTop")
 - FLAG_ACTIVITY_CLEAR_TOP → Destruye todas las actividades que estén por encima en la pila y entonces se comporta como "singleTop"
 -

```
Intent i = new Intent(getApplicationContext(),B.class);  
i.setFlags (Intent.FLAG_ACTIVITY_NEW_TASK);  
startActivity(i);
```



CONTEXT E INTENTS

○ La pila de actividades

- Cuando una pila de actividades está en background por un tiempo, el sistema operativo elimina todas las actividades menos la de la cabecera
- Se puede modificar este comportamiento a través del manifiesto
 - android:alwaysRetainTaskState="True" → Si está en la actividad de la cabecera, mantiene el estado de la pila
 - android:clearTaskOnLaunch="True" → Si está en la actividad raíz, esta será la única se mantenga en cuanto la pila pase a background
 - android:finishOnTaskLaunch="True" → Destruye la actividad en la que esté puesto en cuanto la pila pase a background



CONTEXT E INTENTS

- **Ejercicio 3 (a):** Cread una aplicación con 3 actividades distintas. Cada actividad debe tener un campo de texto (un EditText) y un botón para llamar a cada una de las otras dos actividades.
 - Navegad por varias actividades e id escribiendo algo distinto en cada una de ellas
 - Ahora id pulsando el botón back hasta que salgáis de la aplicación ¿Qué ocurre?



CONTEXT E INTENTS

- **Ejercicio 3 (b):** Modificad la aplicación para que todas las actividades se lancen en modo `singleInstance`
 - Navegad por varias actividades e id escribiendo algo distinto en cada una de ellas
 - Ahora id pulsando el botón back hasta que salgáis de la aplicación ¿Qué ocurre?



CONTEXT E INTENTS

- **Ejercicio 3 (c):** Modificad la aplicación para que todas las actividades se lancen en modo standard, pero detrás de cada *startActivity(...)* añadid un *finish()*
 - Navegad por varias actividades e id escribiendo algo distinto en cada una de ellas
 - Ahora id pulsando el botón back hasta que salgáis de la aplicación ¿Qué ocurre?

