Desarrollo de apps con Android

2. Activity, Intent y Fragment

Activity

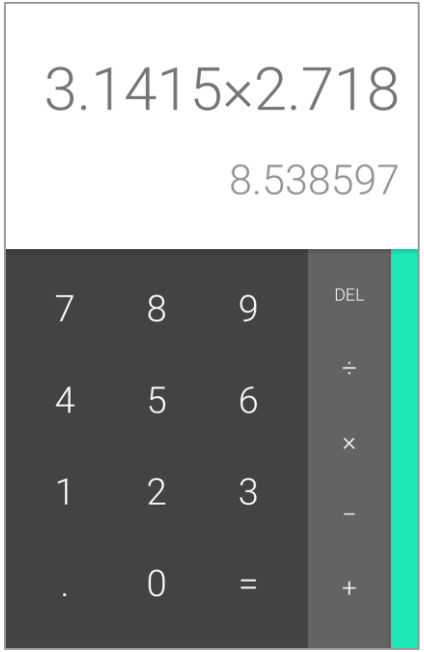
¿Qué es una Activity en Android?

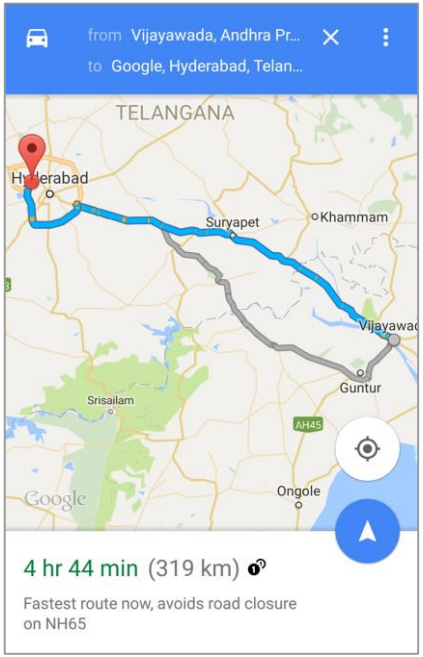
- Una Activity representa una unidad de interacción con el usuario.
- Proporciona la ventana en la que la app dibuja su interfaz de usuario.
- Lo normal es que ocupe toda la pantalla de la app.
- Representa una clase en Java.
- Una app suele estar compuesta de una o varias `Activity` y se puede navegar entre ellas.
- Normalmente la primera `Activity` que se lanza se suele identificar con el nombre ***MainActivity***.

¿Qué se hace en una Activity?

- Representa lo que sería una actividad, como por ejemplo, realizar un pedido o enviar un email.
- Gestiona las interacciones con el usuario, tales como los clics que se realizan sobre un botón o la entrada de texto.
- Puede iniciar otras actividades dentro de la misma app o de otras apps.
- Tiene un ciclo de vida: se crea, se inicia, se ejecuta, se pausa, se reanuda, se para y se destruye.

Ejemplos de Activity





Layout y Activity

- Lo normal es que una `Activity` tenga asociado el layout con el que se tiene que mostrar.
- Cuando se crea la `Activity` se carga (“inflates”) el layout.

Implementación de una Activity

Implementación de una Activity

1. Definir layout en XML.
2. Definir la clase Java con la `Activity`.
    - Hereda de la clase `AppCompatActivity`.
3. Conectar la `Activity` con su Layout .
    - Invocar a `setContentView(layout)` en el método `onCreate()`.
4. Declarar la `Activity` en el fichero **AndroidManifest.xml** (en las últimas versiones de Android Studio las añade automáticamente).

1. Definir layout en XML

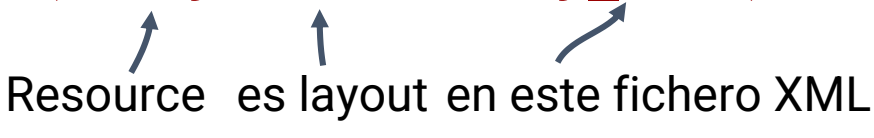
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

2. Definir Activity en una clase Java

```
package es.ucm.fdi.helloworld;
...
public class MainActivity extends AppCompatActivity {
   @Override
   protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
   }
}
```

3. Conectar Activity con layout

```
package es.ucm.fdi.helloworld;

...

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



4. Declarar Activity en AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
...
 <application …>
 ...
    <activity
        android:name=".MainActivity"
        android:exported="true">
 ...
      </activity>
 </application>
</manifest>
```

4. Declarar la Activity de inicio de la app

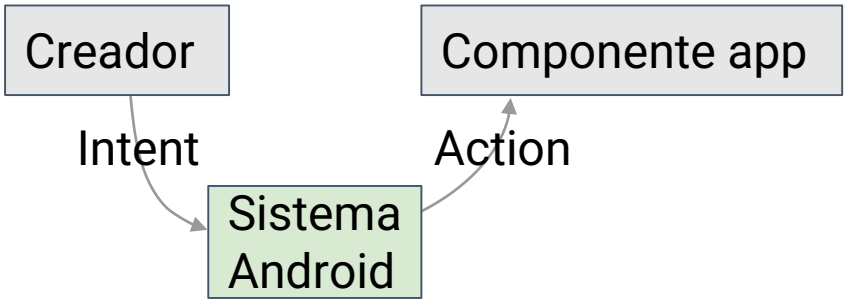
```
<activity
    android:name=".MainActivity"
    android:exported="true">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

Intent

¿Qué es un Intent?

Un Intent describe una operación que se debe llevar a cabo.

Un Intent es un objeto que se utiliza para solicitar una acción desde otro componente de la app a través del sistema de Android.



¿Qué puede hacer un Intent?

- **Iniciar una actividad**
  - El clic sobre un botón inicia una nueva Activity con el texto de entrada.
  - Pulsar en la opción “Share” abre una app que permite publicar una fotografía.
- **Iniciar un servicio**
  - Iniciar la descarga de un fichero en background.
- **Emisiones (*broadcast*)**
  - El sistema informa que el teléfono se está cargando.
- **Proveedores de contenido (*content providers*)**
  - Localización de almacenamiento persistente a la que una app puede acceder.

Intent explícito e implícito

- **Intent explícito:** Inicia una `Activity` específica (ej., MainActivity inicia la `Activity` ViewShoppingCart).
- **Intent implícito:** Solicita al sistema que encuentre una `Activity` que pueda manejar una petición determinada (ej., encontrar una tienda que venda té verde).

Iniciar Activity

Iniciar Activity con un Intent explícito

Para arrancar una `Activity`, utilizar un `Intent` explícito.

1. Crear `Intent`:

   ```
   Intent intent = new Intent(this, ActivityName.class);
   ```

2. Utilizar `Intent` para iniciar la `Activity`:

   ```
   startActivity(intent);
   ```

Iniciar Activity con un Intent implícito

Para solicitar a Android que encuentre una `Activity` para manejar una petición, utilizar un Intent implícito.

1. Crear Intent

```
Intent intent = new Intent(action, uri);
```

2. Utilizar el `Intent` para iniciar la `Activity`:

```
if (intent.resolveActivity(getPackageManager()) != null) {
    startActivity(intent);
}
```

Enviar un Intent implícito con extras

1. Crear un Intent para la acción.

```
Intent intent = new Intent(Intent.ACTION_WEB_SEARCH);
```

2. Incluir los extras.

```
String query = edittext.getText().toString();
intent.putExtra(SearchManager.QUERY, query));
```

3. Iniciar la Activity

```
if (intent.resolveActivity(getPackageManager()) != null) {
    startActivity(intent);
}
```

Intent implícito – Ejemplos (1/3)

Mostrar una página web:

```
Uri uri = Uri.parse("http://www.google.com");
Intent it = new Intent(Intent.ACTION_VIEW,uri);
startActivity(it);
```

Marcar un número de teléfono:

```
Uri uri = Uri.parse("tel:8005551234");
Intent it = new Intent(Intent.ACTION_DIAL, uri);
startActivity(it);
```

Intent implícito – Ejemplos (2/3)

Compartir información por email o redes sociales:

```
Intent sendIntent = new Intent();
sendIntent.setAction(Intent.ACTION_SEND);
sendIntent.putExtra(Intent.EXTRA_TEXT, "This is my text to send.");
sendIntent.setType("text/plain");
Intent shareIntent = Intent.createChooser(sendIntent, null);
startActivity(shareIntent);
```

Intent implícito – Ejemplos (3/3)

Enviar un email:

```
Intent intent = new Intent(Intent.ACTION_SEND);
intent.setType("text/plain");
intent.putExtra(Intent.EXTRA_SUBJECT, "asunto");
intent.putExtra(Intent.EXTRA_TEXT, "texto del email");
intent.putExtra(Intent.EXTRA_EMAIL, new String[] {"user@ucm.es"});
startActivity(intent);
```

Registrar una app para recibir Intent

- Declarar los filtros correspondientes en el fichero **AndroidManifest.xml** de la `Activity`.
- El filtro indica la capacidad de la `Activity` de aceptar un `Intent` implícito.
- El filtro indica condiciones sobre el tipo de `Intent` que acepta la `Activity`.
- Una `Activity` puede contener varios filtros de `Intent`.

Filtro del Intent en AndroidManifest.xml

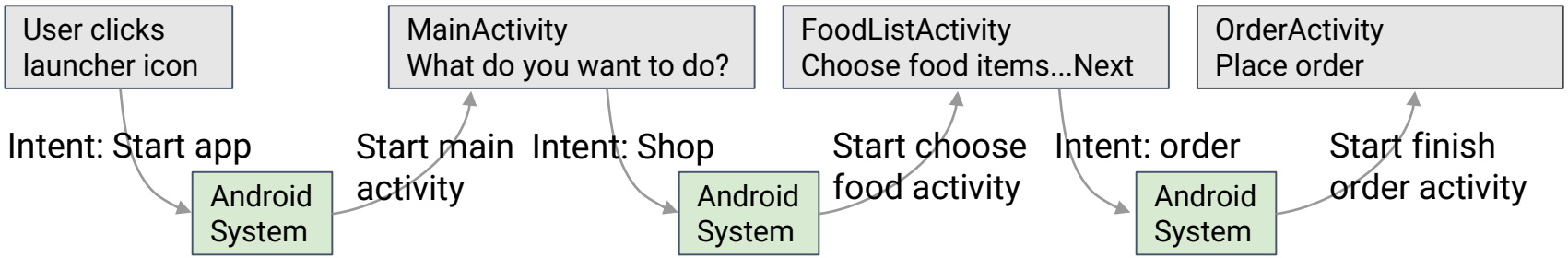
```
<activity android:name="ShareActivity">
  <intent-filter>
    <action android:name="android.intent.action.SEND"/>
    <category android:name="android.intent.category.DEFAULT"/>
    <data android:mimeType="text/plain"/>
  </intent-filter>
</activity>
```

Referencias

- Cómo permitir que otras apps inicien tu actividad
- <intent-filter>
- Intents y filtros de Intents

Ejecución de una Activity

- Todas las instancias de una `Activity` son gestionadas por el runtime de Android.
- Iniciada por un `Intent`, el runtime de Android recibe un mensaje para que ejecute dicha `Activity`.



Envío y recepción de datos

Maneras de enviar datos con Intent

- **Data:** Una porción de información donde la localización de los datos se representa con una URI .
- **Extras:** Una o más porciones de información se envían como una colección de clave-valor en un Bundle.

Envío y recuperación de datos

En la primera `Activity` (la que envía):

1. Crear el objeto `Intent`.
2. Poner data o extras en el `Intent`
3. Iniciar la nueva `Activity` con `startActivity()`

En la segunda `Activity` (la que recibe):

1. Recuperar el objeto `Intent` de la `Activity` con la que se inició.
2. Recuperar data o extras de dicho objeto.

Poner una URI como datos del Intent

```
// URL de una página web.
intent.setData(
    Uri.parse("http://www.google.com"));

// Un fichero URI de muestra.
intent.setData(
     Uri.fromFile(new File("/sdcard/sample.jpg")));
```

Poner información como extras del Intent

- putExtra(String name, int value)

  ⇒ `intent.putExtra("level", 406);`

- putExtra(String name, String[] value)

  ⇒
  ```
  String[] foodList = {"Rice", "Beans", "Fruit"};
  intent.putExtra("food", foodList);
  ```

- putExtras(bundle);

  ⇒ Si hay varios datos, primero crear el bundle y pasarlo como parámetro:

  ```
  Bundle myBundle = new Bundle();
  myBundle.putString("key01", "From Main Activity");
  ...
  intent.putExtras(myBundle);
  ```

Recuperar datos de un Intent

- `getIntent();`
  `⇒ Intent intent = getIntent();`
- `getData();`
  `⇒ Uri locationUri = intent.getData();`
- `int getIntExtra (String name, int defaultValue)`
  `⇒ int level = intent.getIntExtra("level", 0);`
- `Bundle extras = intent.getExtras();`
  `⇒ Recupera todos los datos en un bundle:`

```
String message = extras.getString("key01", "Empty message");
int pos = extras.getInt("key02", 0);
...
```

Devolver datos a la Activity (1/3)

- En la primera `Activity` declarar la variable resultado:

```
private ActivityResultLauncher<Intent> mStartForResult = registerForActivityResult
     (new ActivityResultContracts.StartActivityForResult(),
      new ActivityResultCallback<ActivityResult>() {
              @Override
               public void onActivityResult(ActivityResult result) {
                   switch (result.getResultCode()) {
                       case Activity.RESULT_OK:
                           // Recuperar intent.
                           break;
                       case Activity.RESULT_CANCELED:
                           // Procesar
                           break;
                   }
               }
     });
```

Devolver datos a la Activity (2/3)

- En la primera `Activity`, iniciar la segunda `Activity` de la siguiente manera:

```
Intent intent= new Intent(this, SecondActivity.class);
intent.putExtra("level", "Basic");
mStartForResult.launch(intent);
```

Devolver datos a la Activity (3/3)

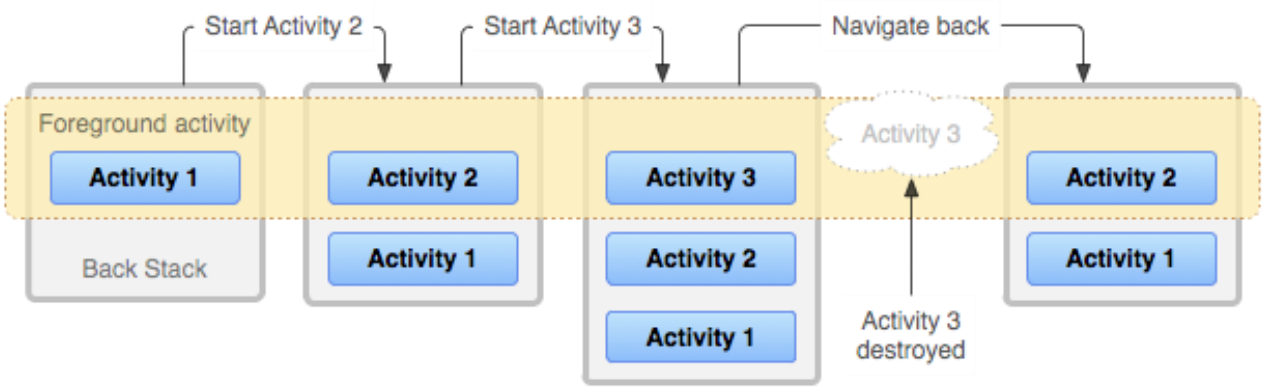
- La segunda `Activity` devuelve el resultado de la siguiente manera:

```
// Crear intent.
Intent replyIntent = new Intent();

// Poner los datos a devolver como extra.
replyIntent.putExtra(EXTRA_REPLY, reply);

// Indicar que el resultado ha sido satisfactorio: RESULT_OK.
setResult(RESULT_OK, replyIntent);

// Dar por terminada la Activity.
finish();
```

Navegación

Activity stack

- Cuando una `Activity` se inicia, la `Activity` anterior se para y se añade a la pila.
- **Last-in-first-out-stack:** Cuando la `Activity` actual termina, o el usuario pulsa el botón Back, se extrae de la pila y se reanuda la `Activity` anterior.



Dos formas de navegación

Temporal o navegación “back”

- Proporcionada por el botón Back del dispositivo.
- Controlada por la pila del sistema Android.

Ancestral o navegación “up”

- Proporcionada por el botón Up de la barra de acción de la app.
- Controlada por las relaciones padre-hijo establecidas en el fichero **AndroidManifest.xml**.

Navegación Back

- La pila Back conserva la historia de las pantallas más recientes.
- La pila Back contiene todas las instancias de `Activity` que han sido lanzadas por el usuario en el orden inverso para **la tarea actual**.
- Cada tarea tiene su propia pila Back.

Navegación Up

- Vuelve al padre de la `Activity` actual.
- Definir el padre de una `Activity` en el fichero **AndroidManifest.xml**.
- Establecer `parentActivityName`:

```
<activity
    android:name=".ShowDinnerActivity"
    android:parentActivityName=".MainActivity" >
</activity>
```

Ciclo de vida Activity

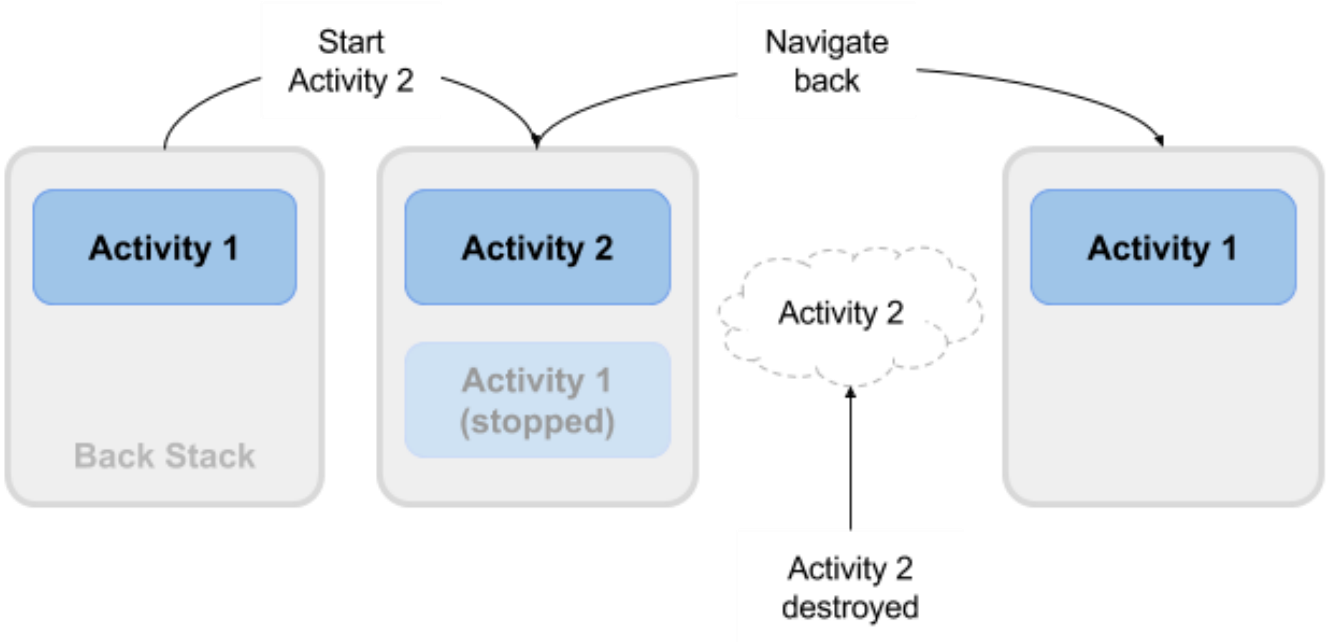
¿Qué es el ciclo de vida de una Activity? (1/2)

- El ciclo de vida de una Activity es el conjunto de estados por los que pasa una `Activity` desde que se crea hasta que se destruye.

De manera más formal:

- Un grafo dirigido de todos los estados en los que puede estar una `Activity`, y los callbacks asociados con la transición de un estado al siguiente.

¿Qué es el ciclo de vida de una Activity? (2/2)



Estados de una Activity y visibilidad app

- Creada (todavía no visible)
- Iniciada (visible)
- Reanudada (visible)
- Pausada (parcialmente visible)
- Parada (oculta)
- Destruida (eliminada de la memoria)

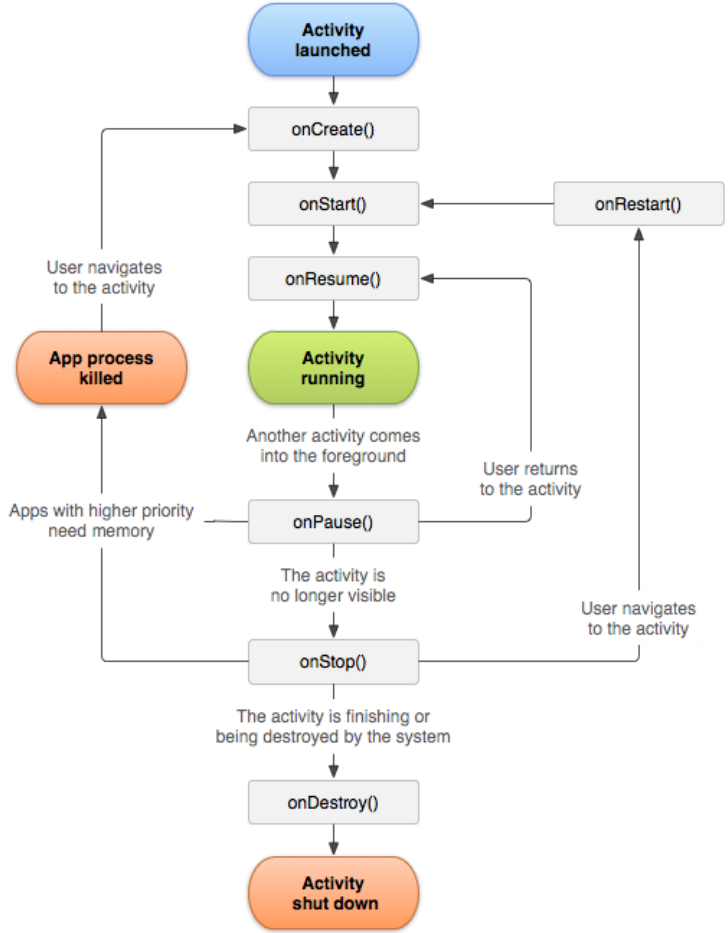
Los cambios de estado se disparan por las acciones de los usuarios, los cambios en la configuración, por ejemplo, al rotar el dispositivo, o por acciones del sistema.

Callbacks del ciclo de vida Activity

Callbacks

- **`onCreate(Bundle savedInstanceState)`**—inicialización estática.
  - **`onStart()`**—cuando `Activity` (es decir, la pantalla) se hace visible.
  - **`onRestart()`**—llamada si la `Activity` se ha parado (llama onStart()).
    - **`onResume()`**—empieza a interactuar con el usuario.
    - **`onPause()`**—reanuda la `Activity` anterior.
  - **`onStop()`**—no está visible, pero todavía existe y se ha conservado la información de su estado.
- **`onDestroy()`**—llamada final antes de que el sistema Android destruya la `Activity`.

Estados y callbacks de una Activity



Implementar y sobrescribir callbacks

- Solamente se requiere el método `onCreate()`.
- Sobrescribir otros callbacks puede alterar el comportamiento por defecto, por ejemplo:

```
@Override
protected void onResume() {
    super.onResume();
    // The activity has become visible
    // it is now "resumed"
}
```

onCreate(Bundle savedInstanceState)

```
@Override
public void onCreate(Bundle savedInstanceState) {
 super.onCreate(savedInstanceState);
 // The activity is being created.
 //setRequestedOrientation(ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
}
```

Estados de la instancia Activity

¿Qué hace que cambie la configuración?

Cambios en la configuración invalidan el layout actual u otros recursos de la actividad cuando el usuario:

- Rota el dispositivo.
- Cambia el idioma del sistema.
- Entra en modo multiventana (a partir de Android 7).

¿Qué ocurre cuándo cambia la configuración?

Ante un cambio de configuración, Android:

1. Da por terminada Activity invocando a:
    - onPause()
    - onStop()
    - onDestroy()
2. Inicia Activity de nuevo invocando a:
    - onCreate()
    - onStart()
    - onResume()

Estado de la instancia de una Activity

- La información de estado se crea mientra la `Activity` está en ejecución, como por ejemplo, el valor de un contador, el texto de un usuario o el progreso de una animación.
- El estado se pierde si se rota el dispositivo, cambia el idioma, se pulsa el botón de retroceder o el sistema limpia memoria.

Guardar y recuperar estado Activity

Qué guarda el sistema

- El sistema solamente guarda:
  - El estado de las `View` que tengan un ID único (`android:id`), como por ejemplo, el texto introducido en un `EditText`.
  - `Intent` que ha iniciado una actividad y los datos que contempla.
- Si se desea guardar otra información ya es responsabilidad de la app (contemplar en el desarrollo).

Guardar el estado de una instancia

Implementar el método `onSaveInstanceState()` en la `Activity`:

- Llamado por el runtime de Android cuando ve posibilidad de que la `Activity` se pueda destruir.
- Guarda los datos de esta instancia de `Activity` solamente para la sesión actual (es decir, mientras la app esté en ejecución).

onSaveInstanceState(Bundle savedInstanceState)

```
@Override
public void onSaveInstanceState(Bundle savedInstanceState)
{
    super.onSaveInstanceState(savedInstanceState);

    mShowCount = findViewById(R.id.show_count);
    savedInstanceState.putString("count",
                String.valueOf(mShowCount.getText()));
}
```

Recuperar el estado de una instancia

Dos formas de recuperar el `Bundle` guardado:

- Método `onCreate(Bundle savedInstanceState)`.
  La mejor opción, para asegurarse que la interfaz del usuario vinculada a la `Activity` incluyendo su estado que ha sido guardado se restaura y se recupera lo antes posible.
- Implementar callback (llamado después del método `onStart()`):
  onRestoreInstanceState(Bundle savedInstanceState)

Recuperación de estado en onCreate()

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    mShowCount = findViewById(R.id.show_count);

    if (savedInstanceState != null) {
        String count = savedInstanceState.getString("count");
        if (mShowCount != null)
            mShowCount.setText(count);
    }
}
```

onRestoreInstanceState(Bundle state)

```
@Override
public void onRestoreInstanceState (Bundle savedInstanceState) {
   super.onRestoreInstanceState(savedInstanceState);

   if (savedInstanceState != null) {
        String count = savedInstanceState.getString("count");
        if (count != null)
            mShowCount.setText(count);
    }
}
```

Fragment

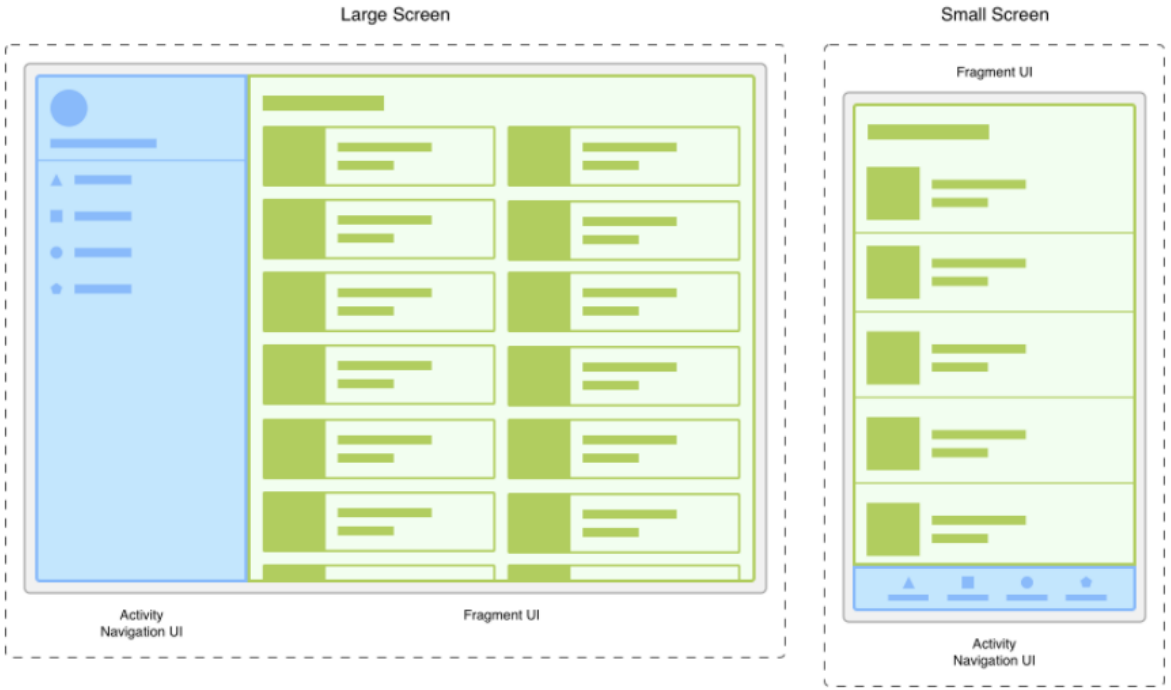
¿Qué es un Fragment? (1/2)

- Un Fragment representa un comportamiento o una parte de la interfaz de usuario que se puede incluir en una `Activity`.
- Por tanto, en este escenario una `Activity` se convierte en un contenedor donde se han embebido uno o más elementos `Fragment`.
- Por tanto, en una misma `Activity` se pueden combinar uno o más `Fragment`.

¿Qué es un Fragment? (2/2)

- Se puede añadir o eliminar en tiempo de ejecución de la `Activity` de la que forma parte.
- Se puede reutilizar en más de una Activity.
- Al igual que una `Activity`, un `Fragment` tiene su propio ciclo de vida y recibe sus propios eventos de entrada.
- Sin embargo, si la `Activity` de la que forma parte se para, los elementos `Fragment` que contenga no se pueden iniciar, y si la `Activity` se destruye también se destruyen todos sus `Fragment`.

Ejemplo de Fragment (1/2)



Ejemplo de Fragment (2/2)