

|  |  |
| --- | --- |
| **Nombre del alumno:** Luis Fernando Cayeros Camacho. | **Matricula:** 2831953. |
| **Módulo 1** |  |
| **Fecha: 18/08/2021**  **Nombre del profesor: Carolina De La Torre Hernández** | Título: Actividad 1 **Blbliografia:**  Cómo interpretar el ciclo de vida de una actividad. (2020). Retrieved 19 August 2020, from https://developer.android.com/guide/components/activities/activity-lifecycle?hl=es#kotlin |

1. **Investiga en**[**developer.android.com**](javascript:;)**cada uno de los métodos definidos en el ciclo de vida de una actividad, describiendo cuando se usa.**onCreate()

“Debes implementar esta devolución de llamada, que se activa cuando el sistema crea la actividad por primera vez. Cuando se crea la actividad, esta entra en el estado Created. En el método onCreate(), ejecutas la lógica de arranque básica de la aplicación que debe ocurrir una sola vez en toda la vida de la actividad. Por ejemplo, tu implementación de onCreate() podría vincular datos a listas, asociar la actividad con un ViewModel y crear instancias de algunas variables de alcance de clase. Este método recibe el parámetro savedInstanceState, que es un objeto Bundle que contiene el estado ya guardado de la actividad. Si la actividad nunca existió, el valor del objeto Bundle es nulo.”  
  
“Si tienes un componente que prioriza el ciclo de vida y que está conectado al ciclo de vida de tu actividad, recibirá el evento ON\_CREATE. Se llamará al método anotado con @OnLifecycleEvent para que tu componente que prioriza el ciclo de vida pueda realizar cualquier código de configuración que necesite para el estado de creación.”  
  
onStart()

“Cuando la actividad entra en el estado Started, el sistema invoca esta devolución de llamada. La llamada onStart() hace que el usuario pueda ver la actividad mientras la app se prepara para que esta entre en primer plano y se convierta en interactiva. Por ejemplo, este método es donde la app inicializa el código que mantiene la IU.

Cuando la actividad pase al estado Started, cualquier componente que priorice el ciclo de vida vinculado al de la actividad recibirá el evento ON\_START.

El método onStart() se completa muy rápido y, al igual que con el estado Created, la actividad no permanece en el estado Started. Una vez finalizada esta devolución de llamada, la actividad entra en el estado Resumed, y el sistema invoca el método onResume().”  
  
onResume()

“Cuando la actividad entra en el estado Resumed, pasa al primer plano y, a continuación, el sistema invoca la devolución de llamada onResume(). Este es el estado en el que la app interactúa con el usuario. La app permanece en este estado hasta que ocurre algún evento que la quita de foco. Tal evento podría ser, por ejemplo, recibir una llamada telefónica, que el usuario navegue a otra actividad o que se apague la pantalla del dispositivo.

Cuando se reanude la actividad, cualquier componente que priorice el ciclo de vida vinculado al de la actividad recibirá el evento ON\_RESUME. Aquí es donde los componentes del ciclo de vida pueden habilitar cualquier funcionalidad que necesite ejecutarse mientras el componente esté visible y en primer plano, como, por ejemplo, iniciar una vista previa de la cámara.

Cuando se produce un evento de interrupción, la actividad entra en el estado Paused y el sistema invoca la devolución de llamada onPause().

Si la actividad regresa al estado Resumed desde Paused, el sistema volverá a llamar al método onResume(). Por esta razón, debes implementar onResume() para inicializar los componentes que lances en onPause() y tienes que realizar otras inicializaciones que deban ejecutarse cada vez que la actividad entre en el estado Resumed.”  
  
onPause()

“El sistema llama a este método a modo de primera indicación de que el usuario está abandonando tu actividad (aunque no siempre significa que está finalizando la actividad); esto indica que la actividad ya no está en primer plano (aunque puede seguir siendo visible si el usuario está en el modo multiventana). Utiliza el método onPause() para pausar o ajustar las operaciones que no deben continuar (o que deben continuar con moderación) mientras Activity se encuentra en estado Paused y que esperas reanudar en breve. Hay varias razones por las que una actividad puede entrar en este estado. Por ejemplo:

Algunos eventos interrumpen la ejecución de la app, como se describe en la sección onResume(). Este es el caso más común.

En Android 7.0 (API nivel 24) o versiones posteriores, varias apps se ejecutan en el modo multiventana. Debido a que solo una de las apps (ventanas) tiene foco en cualquier momento, el sistema pausa todas las demás.

Se abre una nueva actividad semitransparente (como un diálogo). Mientras la actividad siga siendo parcialmente visible, pero no esté en foco, se mantendrá pausada.

Cuando la actividad pase al estado de pausa, cualquier componente que priorice el ciclo de vida vinculado al ciclo de vida de la actividad recibirá el evento ON\_PAUSE. Aquí es donde los componentes del ciclo de vida pueden detener cualquier funcionalidad que no necesite ejecutarse mientras el componente no esté en primer plano, como detener una vista previa de la cámara.

También puedes utilizar el método onPause() para liberar recursos del sistema, controladores de sensores (como el GPS) o cualquier otro recurso que pueda afectar la duración de la batería mientras tu actividad esté en pausa y el usuario no los necesite. Sin embargo, como se mencionó antes en la sección onResume(), una actividad con el estado Paused puede ser completamente visible si está en el modo multiventana. Por eso, deberías considerar usar onStop() en lugar de onPause() para liberar o ajustar por completo los recursos y operaciones relacionados con la IU a fin de admitir mejor el modo multiventana.”  
  
onStop()

“Cuando el usuario ya no puede ver tu actividad, significa que ha entrado en el estado Stopped, y el sistema invoca la devolución de llamada onStop(). Esto puede ocurrir, por ejemplo, cuando una actividad recién lanzada cubre toda la pantalla. El sistema también puede llamar a onStop() cuando haya terminado la actividad y esté a punto de finalizar.

Cuando la actividad pase al estado Stopped, cualquier componente que priorice el ciclo de vida vinculado al de la actividad recibirá el evento ON\_STOP. Aquí es donde los componentes del ciclo de vida pueden detener cualquier funcionalidad que no necesite ejecutarse mientras el componente no sea visible en la pantalla.

En el método onStop(), la app debe liberar o ajustar los recursos que no son necesarios mientras no sea visible para el usuario. Por ejemplo, tu app podría pausar animaciones o cambiar de actualizaciones de ubicación detalladas a más generales. Usar onStop() en lugar de onPause() garantiza que continúe el trabajo relacionado con la IU, incluso cuando el usuario esté viendo tu actividad en el modo multiventana.”  
  
onDestroy()

“  
Se llama a onDestroy() antes de que finalice la actividad. El sistema invoca esta devolución de llamada por los siguientes motivos:

La actividad está terminando (debido a que el usuario la descarta por completo o a que se llama a finish()).

El sistema está finalizando temporalmente la actividad debido a un cambio de configuración (como la rotación del dispositivo o el modo multiventana).

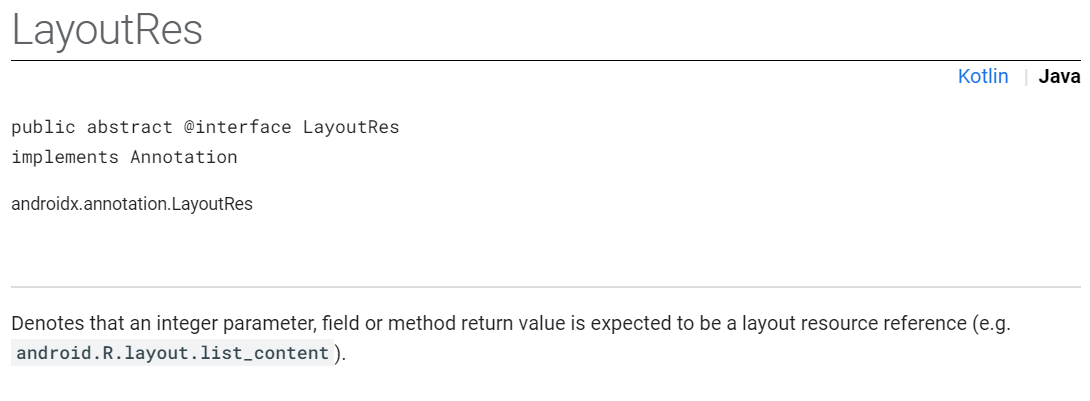
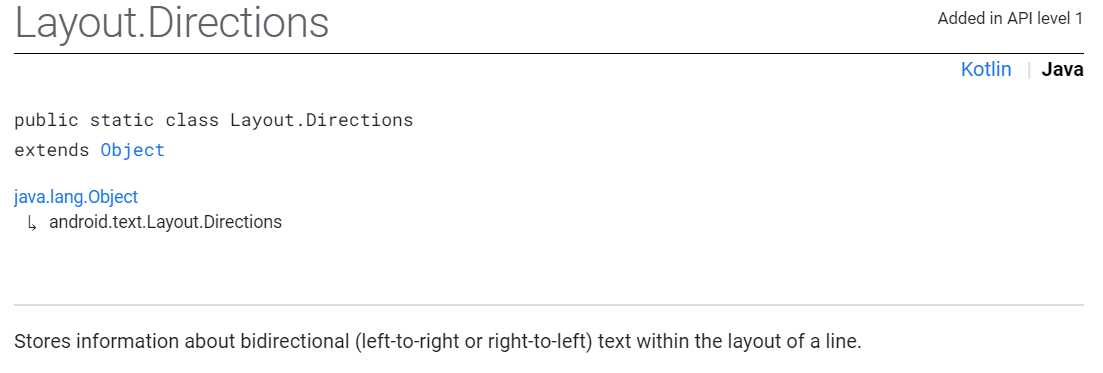
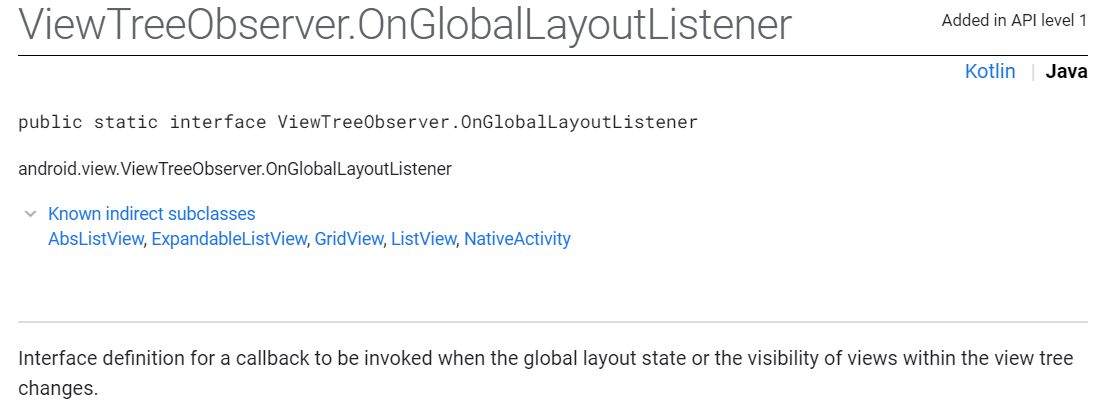
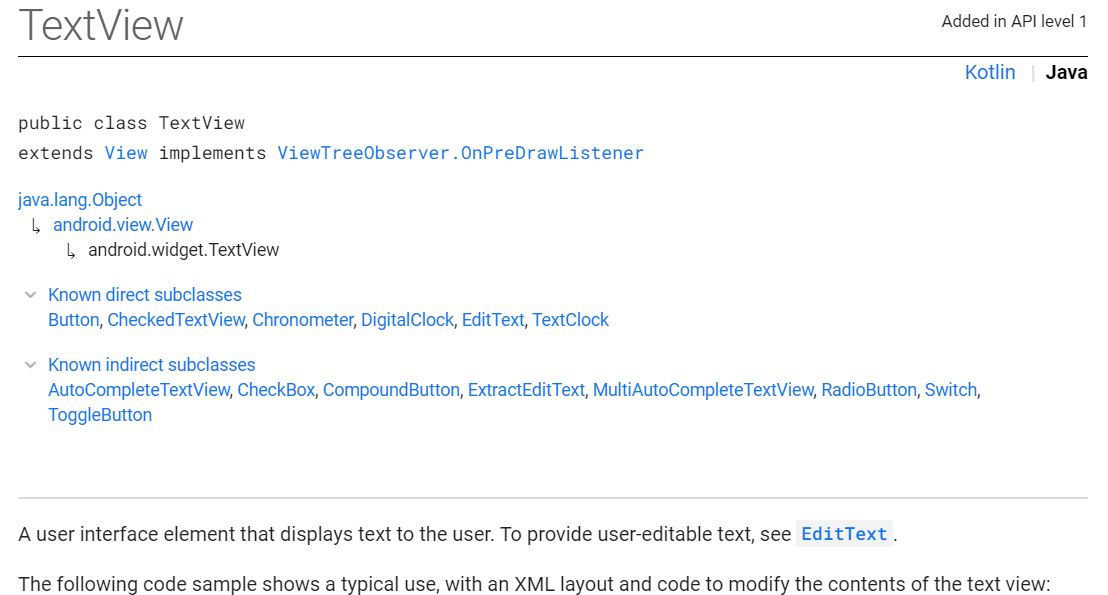
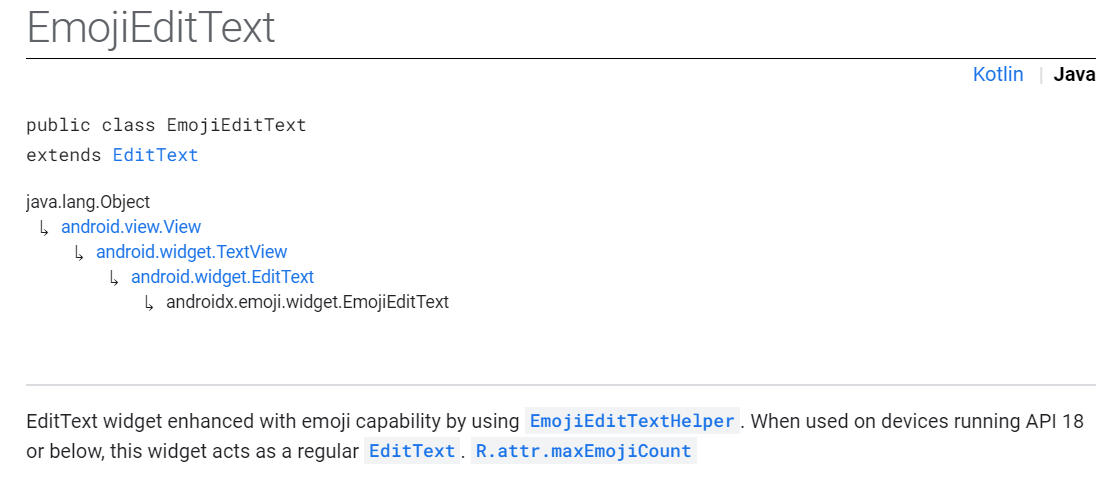
Cuando la actividad pase al estado Destroyed, cualquier componente que priorice el ciclo de vida vinculado al de la actividad recibirá el evento ON\_DESTROY. Aquí es donde los componentes del ciclo de vida pueden recuperar cualquier elemento que se necesite antes de que finalice el objeto Activity.

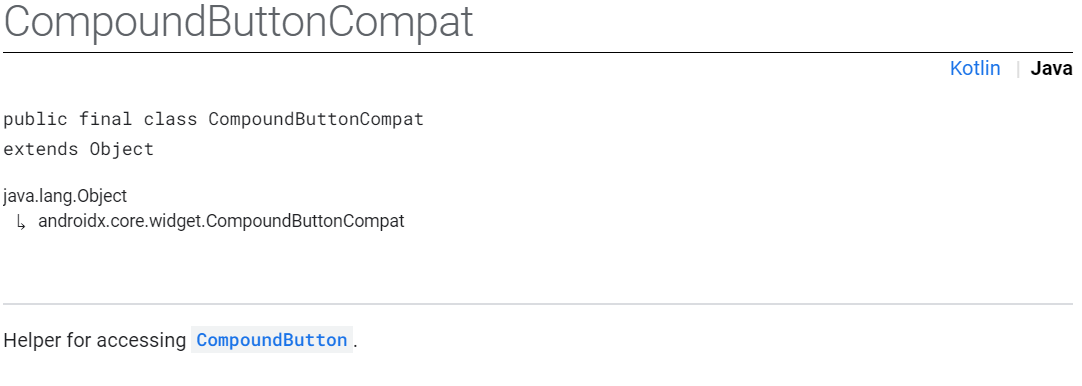
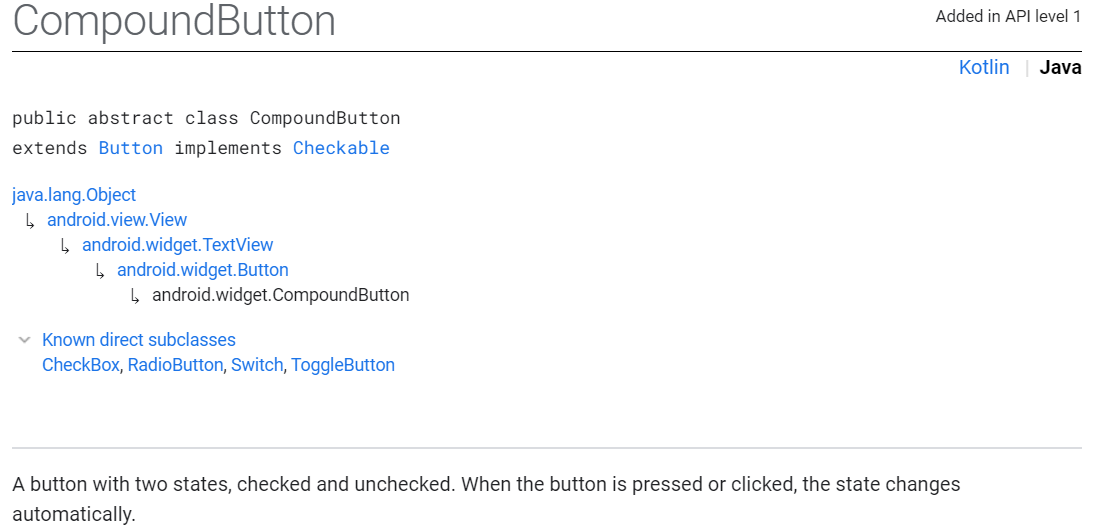
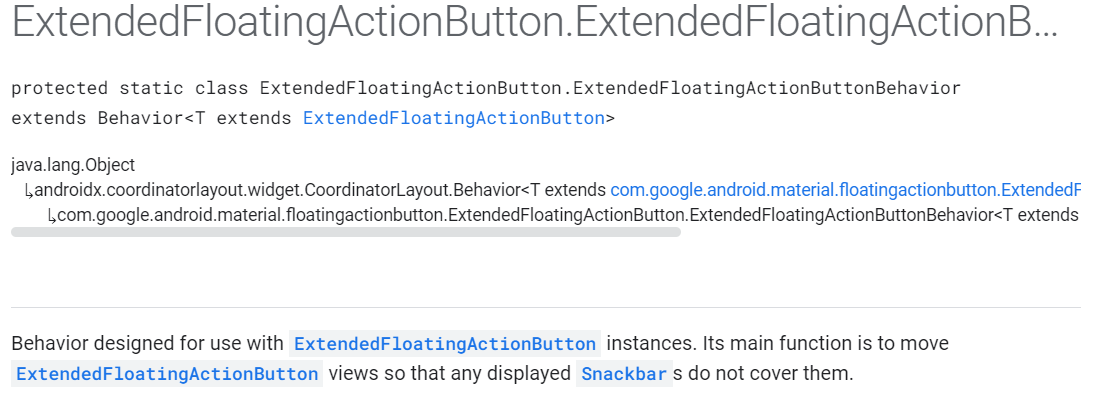
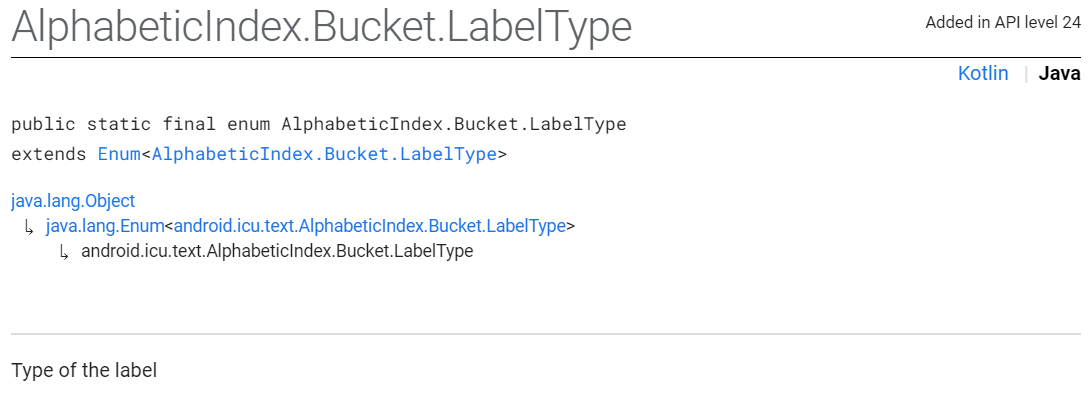
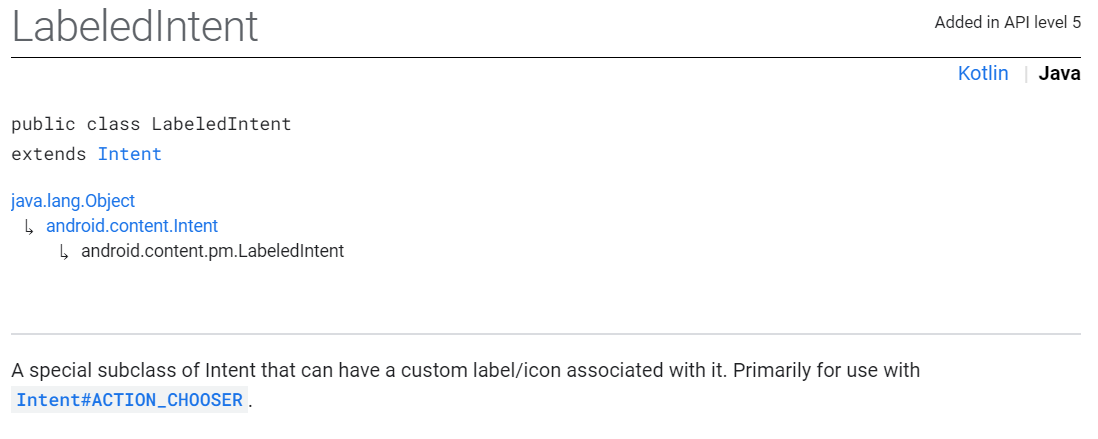
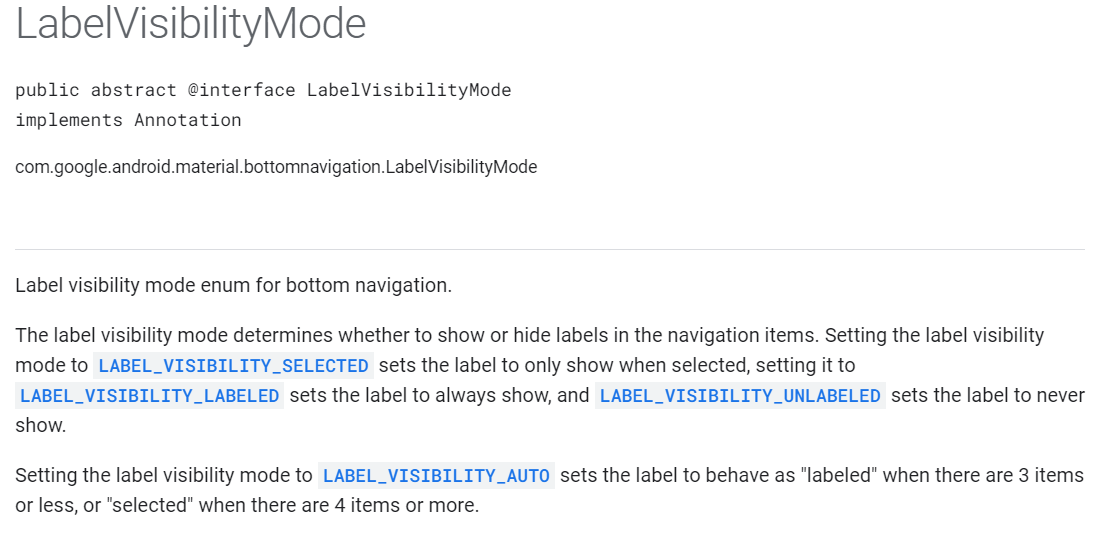
En lugar de poner lógica en ese objeto para determinar por qué está finalizando la actividad, deberías utilizar un objeto ViewModel a fin de contener los datos de vista relevantes para Activity. Si se va a recrear el objeto Activity debido a un cambio de configuración, no es necesario que ViewModel realice ninguna acción, ya que se conservará y se entregará a la siguiente instancia del objeto Activity. Si no se va a recrear el objeto Activity, entonces ViewModel tendrá el método onCleared(), en el que podrá recuperar cualquier dato que necesite antes de que finalice la actividad.

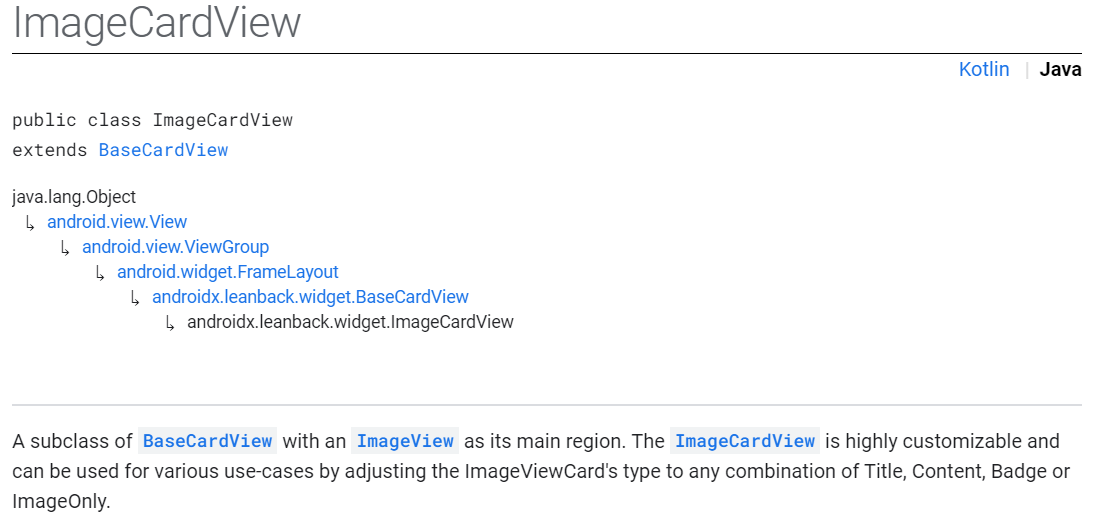
Puedes diferenciar estos dos casos con el método isFinishing().

Si la actividad está terminando, onDestroy() es la devolución de llamada del ciclo de vida final que recibe la actividad. Si se llama a onDestroy() como resultado de un cambio de configuración, el sistema crea inmediatamente una nueva instancia de actividad y luego llama a onCreate() en esa nueva instancia en la nueva configuración.

La devolución de llamada onDestroy() debe liberar todos los recursos que aún no han sido liberados por devoluciones de llamada anteriores, como onStop().”

1. **En el mismo sitio deberás investigar el nombre de la clase y el nombre y descripción de al menos tres métodos para los siguientes elementos de interfaz gráfica:**
   1. **Layout**
   2. **Texto Editable**

* 1. **Botón**
  2. **Etiqueta**

* 1. **Imagen**



