Programación multimedia y dispositivos móviles

6

Ciclo de vida de las actividades

IES Nervión
Miguel A. Casado Alías

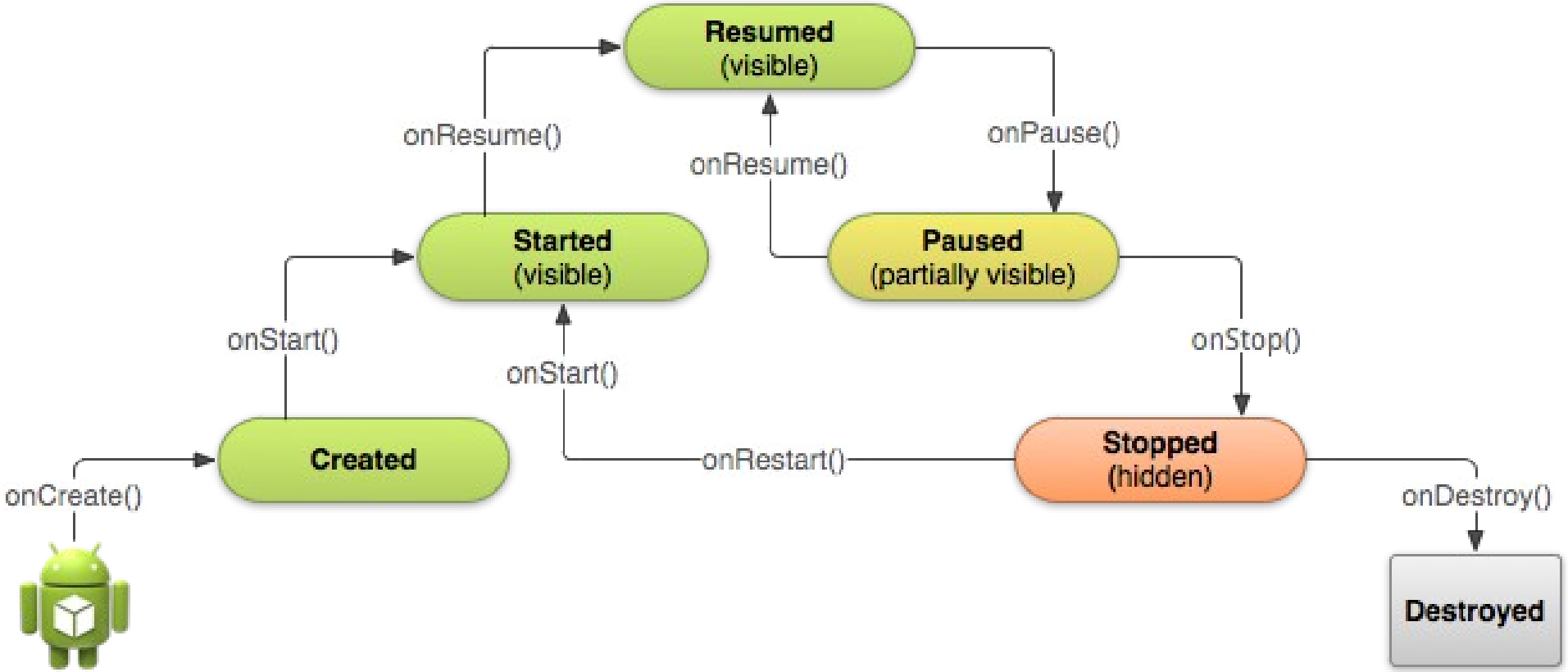
Estados de una actividad

- Una actividad puede estar en tres estados estables:
  - Activa (resumed): La actividad es visible y el usuario puede interactuar con ella
  - Pausada: La actividad está parcialmente tapada por otra actividad semi-transparente o diálogo que está en primer plano. Nuestra actividad, aunque parcialmente visible, no puede interactuar con el usuario ni ejecutar código.
  - Parada: La actividad no se ve, ha pasado a un segundo plano. La instancia de la actividad y su información de estado (valores de atributos por ejemplo) se conservan, pero no se ejecuta código.
- Los estados “Creada” y “Comenzada” son de transición, la actividad permanece en esos estados muy poco tiempo y cambia rápidamente hasta llegar a un estado estable.

Métodos del ciclo de vida

- Para hacer que una actividad pase de un estado a otro, el sistema hace llamadas a una serie de métodos.
- Por ejemplo, tras llamar al método “onCreate” de nuestra actividad, ésta pasará al estado “Creada”.
- O por ejemplo, antes de pausar nuestra actividad, se haría una llamada al método “onPause”.
- Dependiendo de la complejidad de nuestra actividad, es posible que no sea necesario implementar todos los métodos del ciclo de vida. Pero habrá que implementar aquellos necesarios para que la actividad se comporte de la forma en que el usuario espera.

Esquema del ciclo de vida de una actividad



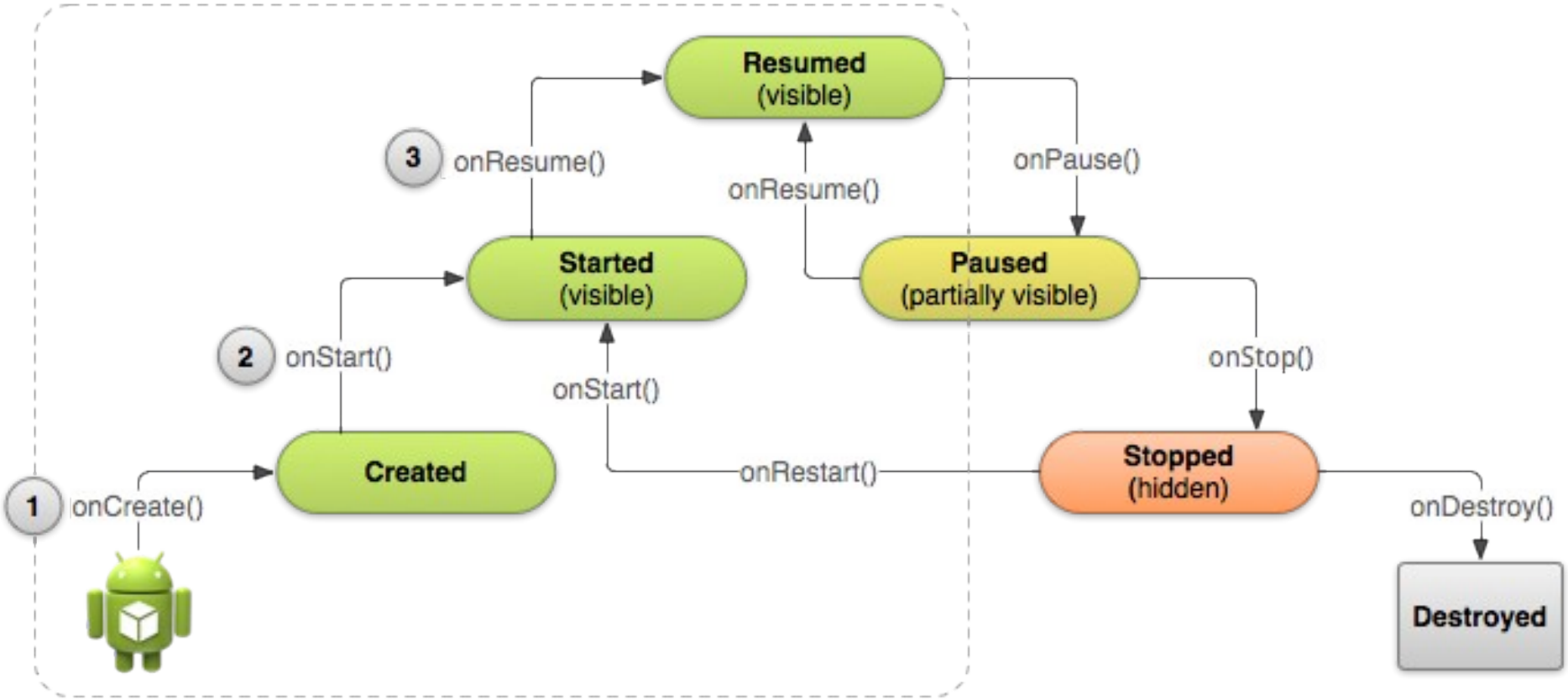
Ventajas de una buena implementación de métodos del ciclo de vida

- La actividad no “petará” si el usuario recibe una llamada o cambia a otra aplicación
- Se liberarán recursos (GPS, cámara, ancho de banda, memoria, etc...) cuando el usuario no está usando activamente la aplicación
- Conservará el progreso del usuario si deja momentáneamente la actividad y vuelve a ella más tarde
- No perderá el progreso del usuario si éste gira la pantalla y pasa de visualizarla en vertical a horizontal (o viceversa)

Creación de una nueva instancia de la actividad

- Cuando el sistema tiene que crear una nueva instancia de una actividad siempre llama a su método **onCreate**
- En la implementación de onCreate se deben realizar aquellas tareas necesarias para “arrancar” la actividad y que sólo deberían ejecutarse una vez en todo el ciclo de vida de la actividad. Por ejemplo, se debería definir la interfaz de usuario y posiblemente instanciar algunas propiedades (atributos) de la clase
- Cuando se acaba de ejecutar onCreate, el sistema llama a **onStart** y seguidamente a **onResume**. Estos métodos son especialmente útiles a la hora de retornar desde los estados de “Pausa” y de “Parada”
- onStart es llamado cuando la actividad comienza a ser visible para el usuario. OnResume es llamado cuando el usuario puede empezar a interactuar con la actividad

Esquema de creación de una actividad



Destrucción de la actividad

- El sistema llama a **onDestroy** justo antes de que la actividad sea totalmente eliminada de la memoria
- Pocas veces es necesario implementar este método, porque las principales tareas de liberación de recursos se harán en las implementaciones de onPause y onStop
- Si nuestra actividad ha creado hilos/tareas en segundo plano, que no destruimos cuando la actividad está pausada o parada, deberíamos eliminarlos en onDestroy
- Sólo hay una situación en la que se llama a onDestroy sin haber llamado antes a onPause y onStop: Cuando nosotros mismos llamamos al método **finish** desde el método onCreate

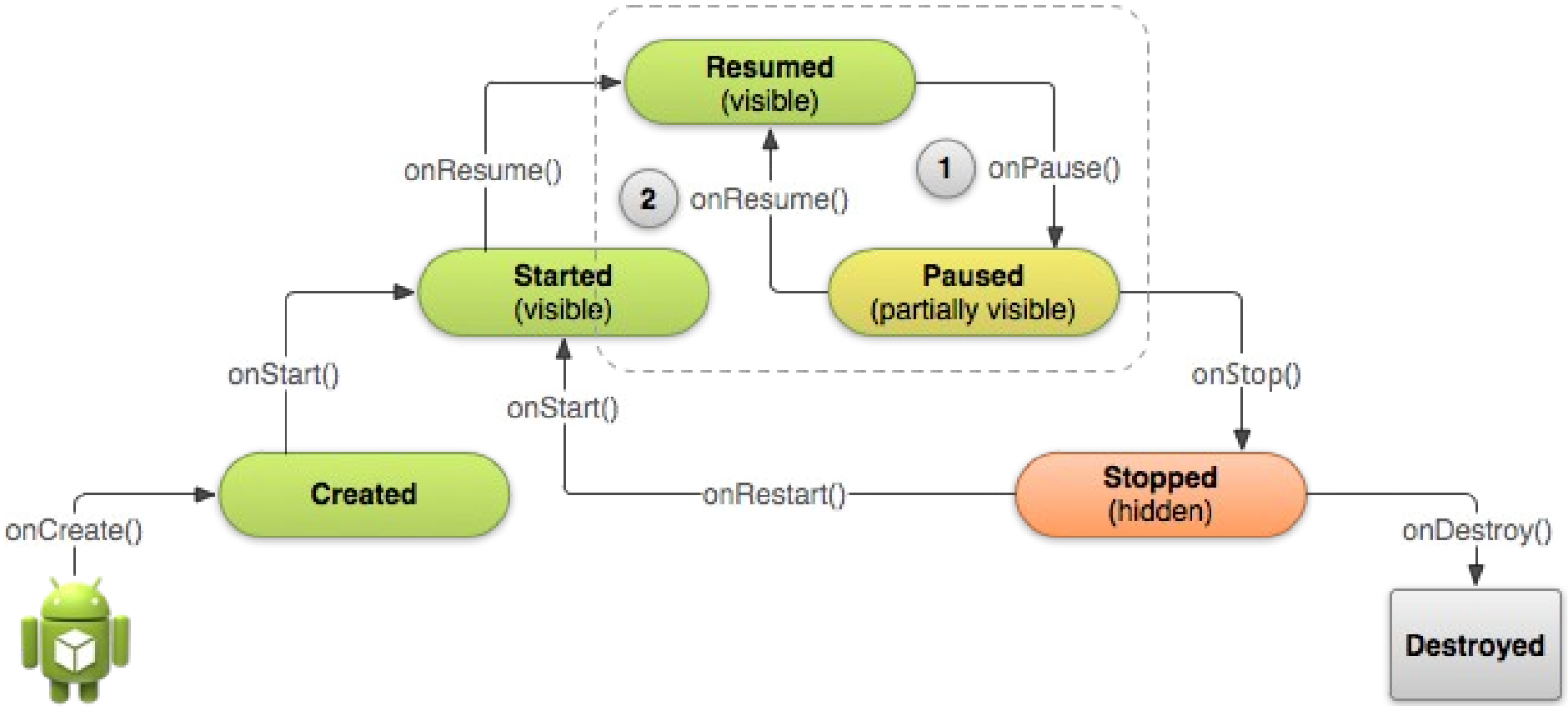
Actividad en pausa

- Una actividad puede entrar en pausa porque un diálogo la oscurece en parte, o porque la actividad se encamina hacia el estado de parada
- Al entrar en pausa, el sistema llama al método **onPause,** y ahí deberemos:
  - Parar acciones en marcha (videos, música, etc...)
  - Guardar información que el usuario esperaría que se guardara (por ejemplo, borrador de un email)
  - Liberar recursos (broadcast receivers, sensores...)
- No se deben realizar en onPause tareas costosas en términos de CPU (p.ej: Guardar en BD) porque puede hacer que la transición hacia la otra actividad (p.ej: el diálogo que aparece en primer plano) sea muy lenta

onResume

- Ése es el método al que el sistema llama **cada vez que la actividad se pone en primer plano**, ya sea cuando se crea por primera vez la actividad o cuando se retorna del estado de pausa
- En onResume se deben inicializar los componentes que se liberaron en onPause (p.ej: cámara, GPS...), y llevar a cabo cualquier otra inicialización que deba ocurrir cada vez que la actividad entra en el estado “Activo” (p.ej: continuar reproducción multimedia)

Esquema de pausa de una actividad



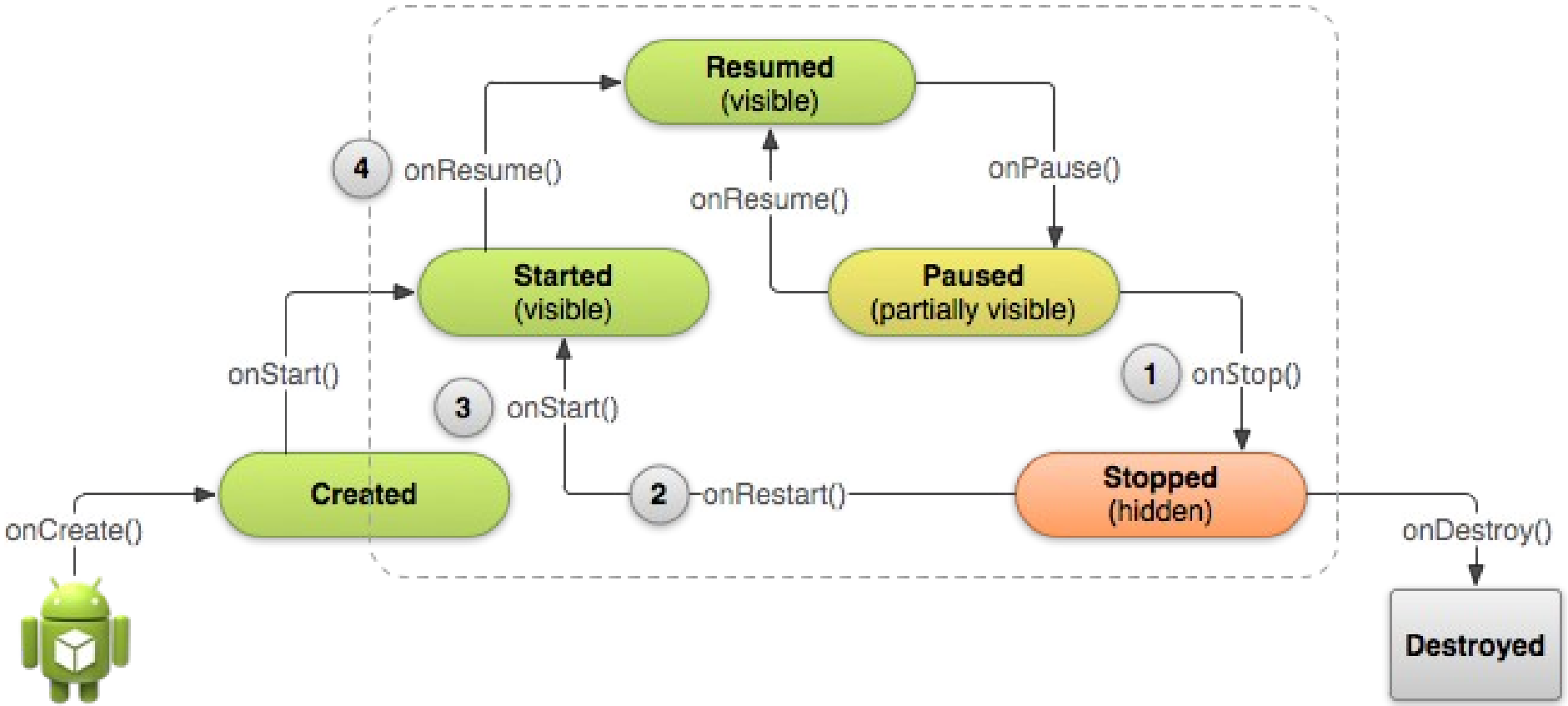
Parada de una actividad

- El sistema llama al método **onStop** antes de parar una actividad. Una actividad se puede parar porque se reciba una llamada, porque el usuario deje la aplicación para pasar a usar otra, porque una acción del usuario provoque que se cree otra actividad dentro de la aplicación...
- Si algún recurso no se liberó durante onPause, se debe liberar en onStop. También se deben llevar a cabo en onStop operaciones de cierre y guardado demasiado intensivas en uso de CPU como para realizarse en onPause
- En casos extremos, el sistema puede “matar” una actividad que está parada sin llamar siquiera a onDestroy

Reinicio de una actividad parada

- Cuando la actividad vuelve al primer plano, el sistema llama a su método **onRestart** y seguidamente a **onStart**
- Como onRestart sólo es llamado después de que la actividad haya sido parada, en él podemos desarrollar cualquier trabajo de recuperación que sea necesario hacer en el caso de que la actividad sea parada pero no destruida
- Es raro que una actividad necesite implementar onRestart
- Como contrapartida a onStop se debe usar onStart. En onStart se reinicializarán aquellos componentes liberados en onStop. **Tener en cuenta además que onStart también es llamado cuando la actividad se crea por primera vez**

Esquema de parada de una actividad



Volviendo a crear una actividad

- Cuando una actividad es destruida por el sistema debido a que hacen falta recursos o a que lleva mucho tiempo sin usarse, **guarda el estado de la instancia** (*instance state*), para que cuando el usuario quiera volver a la actividad destruida, se pueda crear otra instancia de la misma, en la que se puedan cargar los datos salvados previamente, los cuales describen el estado de la actividad cuando fue destruida
- El estado de la instancia es una colección de pares “clave-valor” (*key-value*) almacenados en un objeto Bundle
- Por defecto, el sistema guarda la información de estado de cada View de la actividad en el Bundle mencionado (siempre y cuando el View tenga un id único definido)

onSaveInstanceState

- El sistema siempre llama a este método antes de destruir una actividad
- Este método es el encargado de guardar el estado de la actividad
- Si queremos que además de guardar el estado de las vistas (View) guarde otra información de estado (p.ej: puntos que llevaba el usuario en el juego), deberemos redefinir onSaveInstanceState
- Ejemplo de onSaveInstanceState

onRestoreInstanceState

- El sistema pasa el Bundle con la información de estado a los métodos **onRestoreInstanceState** y **onCreate**
- Como onCreate es llamado tanto si se ha guardado información de estado de una instancia anterior como si no, deberemos comprobar que el Bundle de estado no es *null* antes de leerlo
- En cambio, el método onRestoreInstanceState sólo es llamado si hay información de estado para ser restaurada, y en ese caso es llamado tras onStart
- Podemos hacer la restauración de estado en cualquiera de dichos métodos (onRestoreInstanceState , onCreate)
- Ejemplo de restauración de estado