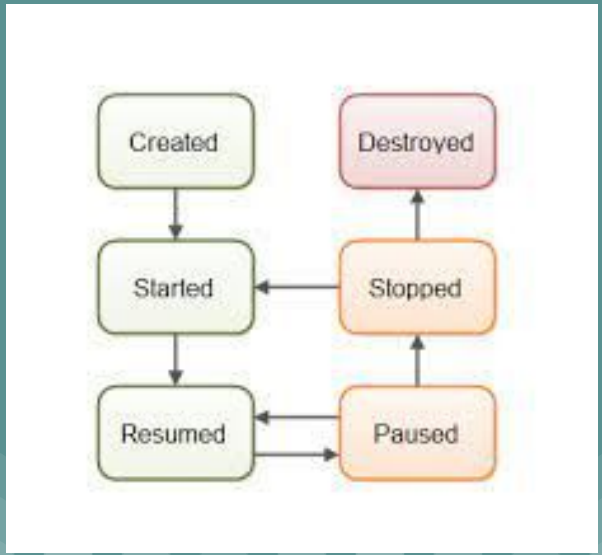
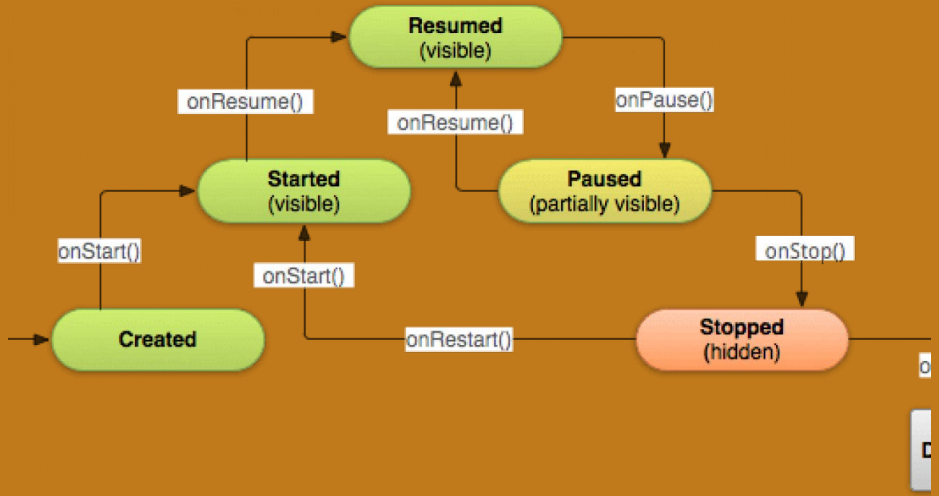
Ciclo de vida de las actividades

Cuando un usuario navega por una app, sale de ella y vuelve a entrar, las instancias de una actividad pasan por diferentes estados de su ciclo de vida. La clase Activity proporciona una serie de callbacks que permiten a la actividad saber que cambió un estado, es decir, que el sistema está creando, deteniendo o reanudando una actividad, o finalizando el proceso en el que se encuentra.



Una buena implementación de los callbacks de un ciclo de vida puede ayudar a garantizar que una app:

- No falle si el usuario recibe una llamada telefónica o cambia a otra app mientras usa la tuya.
- No consuma recursos valiosos del sistema cuando el usuario no la use de forma activa.
- No pierda el progreso del usuario si este abandona tu app y regresa a ella posteriormente.
- No falle ni pierda el progreso del usuario cuando se gire la pantalla entre la orientación horizontal y la vertical.



Para navegar por las transiciones entre las etapas del ciclo de vida de una actividad, la clase Activity proporciona un conjunto básico de seis devoluciones de llamadas: onCreate(), onStart(), onResume(), onPause(), onStop() y onDestroy(). El sistema invoca cada una de estas devoluciones de llamada cuando una operación entra en un nuevo estado.

onCreate

- Debes implementar este callback, que se activa cuando el sistema crea la actividad por primera vez.
- En este método ejecutas la lógica de arranque básica de la aplicación que debe ocurrir una sola vez en toda la vida de la actividad.
- 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.
- La actividad no reside en el estado Created. Después de que se termina de ejecutar el método, entra en el estado Started, y el sistema llama rápidamente a los métodos onStart() y onResume().

```
override fun onCreate(savedInstanceState: Bundle?) {
    // call the super class onCreate to complete the creation of activity like
    // the view hierarchy
    super.onCreate(savedInstanceState)

    // recovering the instance state
    gameState = savedInstanceState?.getString(GAME_STATE_KEY)

    // set the user interface layout for this activity
    // the layout file is defined in the project res/layout/main_activity.xml file
    setContentView(R.layout.main_activity)

    // initialize member TextView so we can manipulate it later
    textView = findViewById(R.id.text_view)
}
```

onStart

- Cuando la actividad entra en el estado Started, el sistema invoca este callback. 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.
- 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 finalizado este callback, 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().
- La app permanece en este estado hasta que ocurre algún evento que la quita de foco.
- 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().

onPause

- El sistema llama a este método a modo de primera indicación de que el usuario está abandonando tu actividad; Esto indica que la actividad ya no está en primer plano.
- Usado para pausar o ajustar las operaciones que no deben continuar mientras Activity se encuentra en estado Paused y que esperas reanudar en breve.
- Razones por las cuales se puede pausar una actividad
  - Algunos eventos interrumpen la ejecución de la app.
  - En Android 7.0 (API 24) o versiones posteriores, varias apps se ejecutan en el modo multiventana. Debido a que solo una de las apps 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.
- La ejecución es muy breve y no necesariamente permite disponer de tiempo suficiente para realizar operaciones seguras.

onStop

- Cuando el usuario ya no puede ver tu actividad, significa que ha entrado en el estado Stopped.
- 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.

```
override fun onStop() {
    // call the superclass method first
    super.onStop()

    // save the note's current draft, because the activity is stopping
    // and we want to be sure the current note progress isn't lost.
    val values = ContentValues().apply {
        put(NotePad.Notes.COLUMN_NAME_NOTE, getCurrentNoteText())
        put(NotePad.Notes.COLUMN_NAME_TITLE, getCurrentNoteTitle())
    }

    // do this update in background on an AsyncQueryHandler or equivalent
    asyncQueryHandler.startUpdate(
            token,      // int token to correlate calls
            null,       // cookie, not used here
            uri,        // The URI for the note to update.
            values,     // The map of column names and new values to apply to them.
            null,       // No SELECT criteria are used.
            null        // No WHERE columns are used.
    )
}
```

- Cuando tu actividad entra en el estado Stopped, se mantiene el objeto Activity en la memoria. Mantiene toda la información de estado y de miembros, pero no está vinculada al administrador de ventanas.
- Desde el estado Stopped, la actividad regresa a interactuar con el usuario o se termina de ejecutar y desaparece. Si la actividad regresa, el sistema invoca a onRestart. Si se terminó de ejecutar, el sistema llamará a onDestroy().

onDestroy

- Se llama antes de que finalice la actividad. El sistema invoca esta devolución de llamada por los siguientes motivos:
  - La actividad está terminando.
  - El sistema está finalizando temporalmente la actividad debido a un cambio de configuración.
- Si se llama 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.

Relación entre el ciclo de vida del proceso y el estado de la actividad

Probabilidad de que finalice	Estado del proceso	Estado de la actividad
Menos	Primer plano (en foco o por estar en él)	Created Started Resumed
Más	Segundo plano (foco perdido)	Pausado
Mayor	Segundo plano (no visible)	Detenido
	Vacío	Finalizado