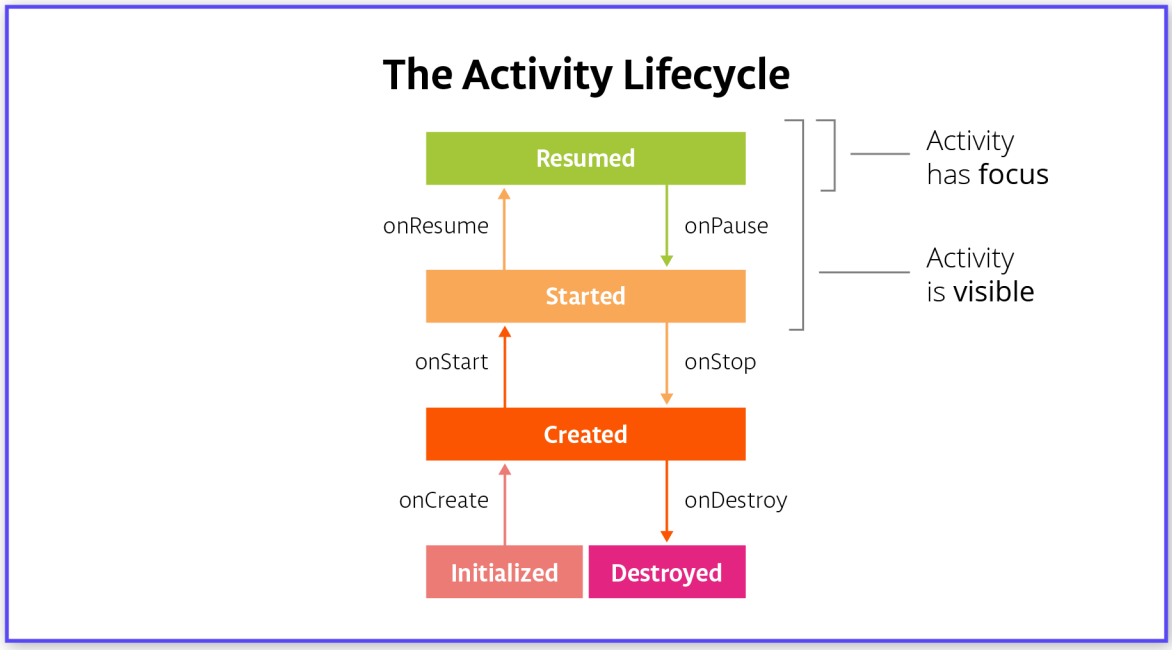
Lesson 4: Activity & Fragment Lifecycle



Activity Lifecycle callback methods

Called when activity moves from one state to antoher.

Perform action when activity enters a certain lifecycle by overriding callback method.

General Defenitions

Visible Lifecycle

The part of the Lifecycle between onStart and onStop when the activity is visible.

Focus

An Activity is said to have focus when it's the activity that the user can iteract with.

Foreground

When the activity is on the screen.

Background

When the activity is fully of screen, it is considered in the background.

Lifecycle States

These are the same for both the Fragment Lifecycle and trhe Activity Lifecycle.

Initialized

This is the starting state whenever you make a new activity. This is a transient state -- it immediately goes to Created.

Created

Activity has just been created, but it's not visible and doesn't have focus. You are not able to interact with it.

Started

Activity is visible but doesn't have focus.

Resumed

The state of the activity when it is running. It's visible and has focus.

Destroyed

Activity is destroyed. It can be ejected from memory at any point and should not be referenced or interacted with.

Activity Lifecycle Callbacks

onCreate

This is called the first time the activity starts, it is only called once during the lifecycle of the activity. It represents when the activity is created and initialized. The activity is not yet visible and you can't interact with it.

You must implement onCreate. You should:

- Inflate the activity's UI, wether that's using findViewById or databinding.
- Initialize variables.
- Do any other initialization that only happens once during the activity lifetime.

onStart

This is triggered when the activity is about to becom visible. It can be called multiple times as the user navigates away from the activity and then back.

In onStart you should:

- Start any sensors, animations or other procedures that need to start when the activity is visible.

onResume

This is triggered when the activity has focus and the user can iteract with it.

Here you should:

- Start any sensors, animations or other procedures that need to start when the activity has focus.

onPause

The mirror method to onResume. Called as soon as the activity loses focus.

Here you should:

- Stop any sensors, animations or other procedures that should not run when the activity doesn't have focus and is partially obscured.
- Keep execution fast. The next activity is not shown untill this completes.

onStop

The mirror method to onStart. It is called when you can no longer see the activity.

Here you should:

- Stop any sensors, animations, or other procedures that should not run when the activity is not on screen.
- You can use this to persist data.
- Stop logic that updates the UI.

onDestroy

The mirrored method to onCreate. Called when the activity is fully destroyed. It is your last chance to clean up resources associated with the activity.

Here you should:

- Tear down or release any resources that are related to the activity and are not automatically released for you. Forgetting to do this can cause a memory leak. Logic that refers to the activity or attempts to update the UI after the activity has been destroyed could crash the app!

Summary of the Fragment Lifecycle

Fragments also have lifecycle states that they go between. The lifecycle states are the same as the activity states. While fragment lifecycle states are the same, the callbacks are different.

Important Fragment Callbacks to Implement

onCreate

Similar to the Activity's onCreate callback. This is when the fragment is created. This wil only get called once.

Here you should:

- Initialize anything essential for your fragment.
- **Do not inflate XML**, do that in onCreateView, when the system is first drawing the fragment NOT reference the activity, it is still being created. Do this in onActivityCreated.

onCreateView

This is called between onCreate and onActivityCreated. You must return a view in this callback if your fragment has a UI.

Here you should:

- Create your views by inflating your XML.

onStop

Very similar to Activity's onStop. This callback is called when the user leaves your fragment.

Here you should:

- Save any permanent fragment state.

Other callbacks

onAttach

When the fragment is first attached to the activity. This is only called once during the lifecycle of the fragment.

onActivityCreated

Called when the activity onCreate method has returned and the activity has been initialized. If the fragment is added to an activity that's already created, this still gets called. It is called multiple times during the lifecycle of the fragment.

Here you should:

- Execute any code that requires an activity instance.

onStart

Called right before the fragment is visible to the user.

onResume

When the activity resumes the fragment. This means the fragment is visible, has focus and is running.

onStop

Called when the activity's onStop is called. The fragment no longer has focus.

onDestroyView

Unlike activities, fragment views are destroyed every time they go off screen. This is called after the view is no longer visible.

onDestroy

Called when Activity's onDestroy is called.

onDetach

Called when the association between the fragment and the activity is destroyed.

Lifecycle Library

Simplifies working with the activity and fragment lifecycle.

Introduces the lifecycle as an actual object. The object can be queried an checked for state.

LifecycleOwner Interface

Class that has a lifecycle. Ex Activity and Fragment. You can ask the owner for it's lifecycle object and get information about the current state of the lifecycle.

LifecycleObserver Interface

Allow objects to observe lifecycle owners.

Process shutdown

The OS can shut down your app when it's in the background. Android tries to to reset your app to the state that it had before. It takes the state of some of your view and saves them to a bundle. When the OS restarts the app, it's going to use this data. You can also manually add data to the bundle with the onSaveInstanceState.

onSaveInstanceState

Callback where you can save data that you might need in case android destroys your app.