Android Development with Kotlin

Android Lifecycle

Coding Ninjas

What is Activity Lifecycle?

As a user navigates through, out of, and back to your app, the Activity instances in your app transition through different states in their lifecycle. The Activity class provides a number of callbacks that allow the activity to know that a state has changed: that the system is creating, stopping, or resuming an activity, or destroying the process in which the activity resides.

Within the lifecycle callback methods, you can declare how your activity behaves when the user leaves and re-enters the activity. For example, if you're building a **streaming video player, you might pause the video and terminate the network connection when the user switches to another app**. When the user returns, you can reconnect to the network and allow the user to resume the video from the same spot. In other words, each callback allows you to perform specific work that's appropriate to a given change of state. Doing the right work at the right time and handling transitions properly make your app more robust and performant.

Coding Ninjas

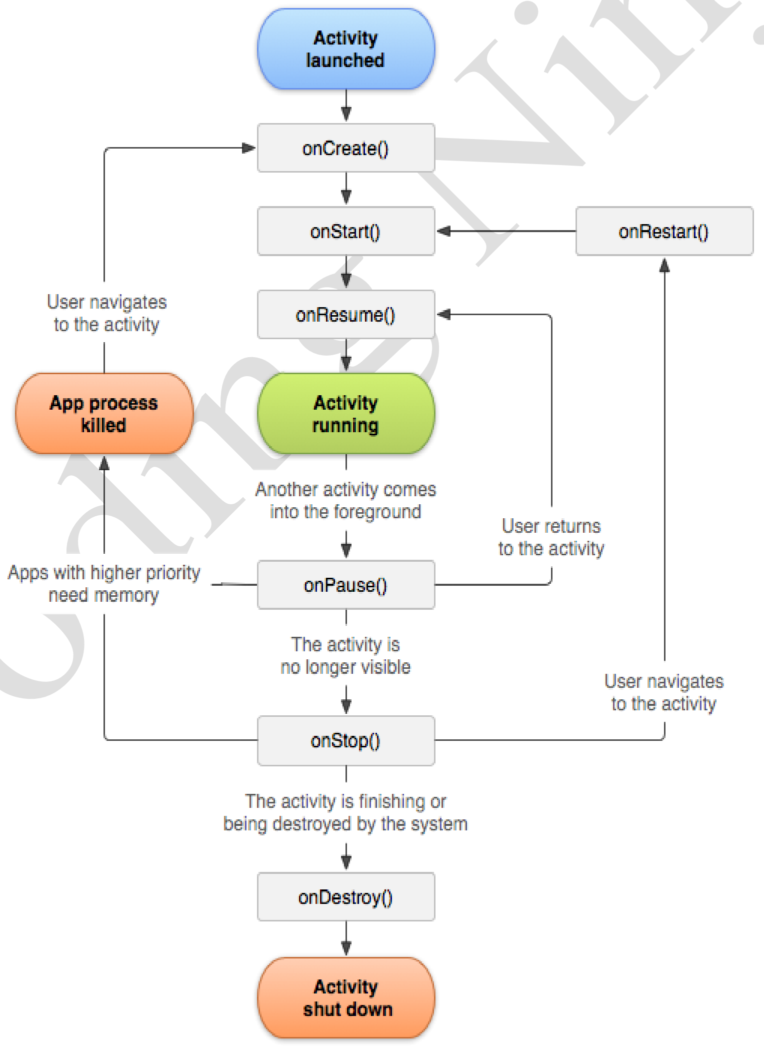

Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

Image src: https://developer.android.com/guide/components/images/activity_lifecycle.png

Following are the activity lifecycle methods with description:

onCreate: This method is called when the activity is created. This is the place where you should call setContentView and set activity’s view. This is where you should set up list’s data and create other views.

onRestart: This is called when an activity is stopped and restarted instead of getting destroyed. This is always followed by onStart.

OnStart: This method is called when the activity starts becoming visible to the user.

onResume: This is when user can start interacting with the activity. Now this activity is in focus.

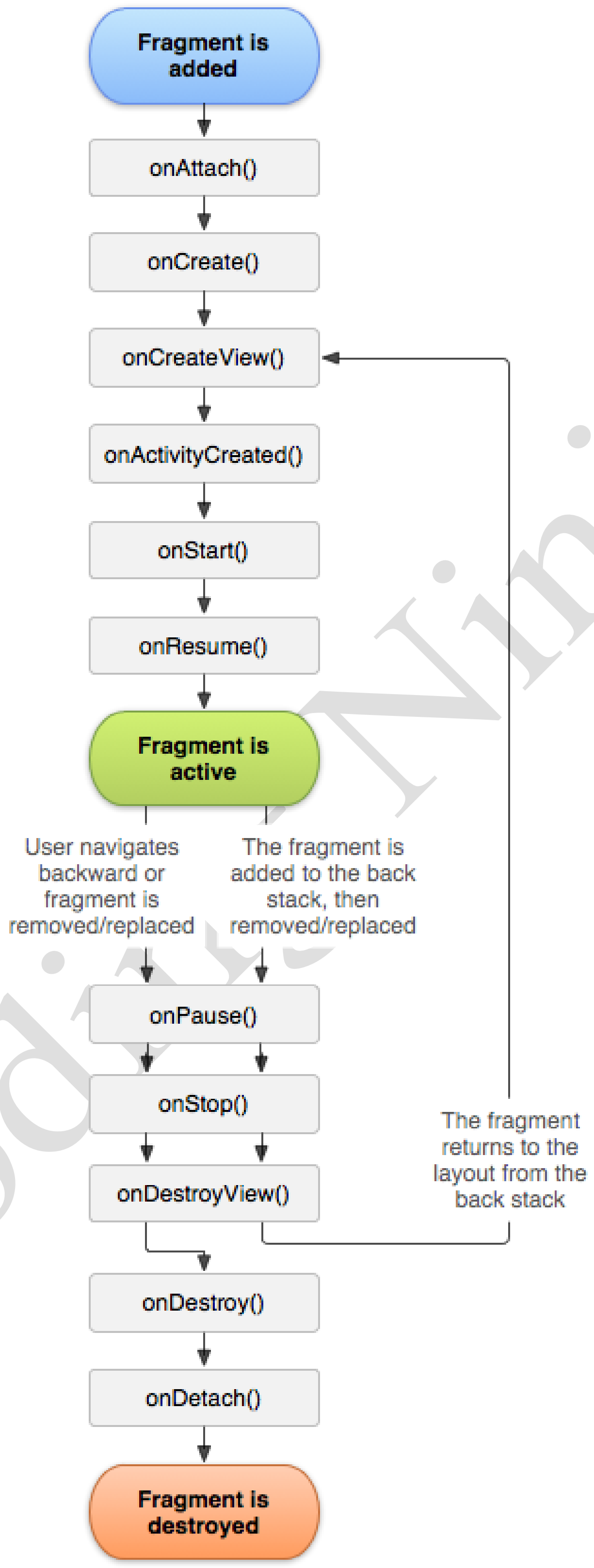
onPause: This function is called when the activity is visible but is loosing focus. E.g. if a new activity is being loaded or a AlertDialog shows up in front of the current activity.

onStop: This is called when the activity is no longer visible to the students. This can happen because a new activity is being created or a previous activity is being pushed to the front.

onDestroy: This is the final call you will receive before the activity is destroyed. An activity can be destroyed either because the system is running low on resources and is killing your activity or the activity has finished. The first condition can only happen if your activity is in background.

Coding Ninjas

Android Fragment Lifecycle:



Coding Ninjas

Following are the lifecycle methods with the description:

onAttach() : The fragment instance is associated with the instance of the enclosing activity. The fragment and the activity is not fully initialized. Typically you get in this method a reference to the activity which uses the fragment for further initialization work.

· **onCreate() -** The system calls this method when creating the fragment. You should initialize essential components of the fragment that you want to retain when the fragment is paused or stopped, then resumed.

· **onCreateView() -** The fragment instance creates its view hierarchy. To draw a UI for your fragment, you must return a **View** component from this method that is the root of your fragment's layout. You can return null if the fragment does not provide a UI (headless fragment). In this method you should not interact with the activity, the activity is not yet fully initialized.

· **onActivityCreated() -** The onActivityCreated() is called after the onCreateView() method when the host activity is created. Activity and fragment instance have been created as well as the view hierarchy of the activity. At this point, view can be accessed with the findViewById() method. In this method you can instantiate objects which require a Context object

· **onStart() -** The onStart() method is called once the fragment gets visible.

· **onResume() -** Fragment becomes active.

· **onPause()** - The system calls this method as the first indication that the user is leaving the fragment. This is usually where you should commit any changes that should be persisted beyond the current user session.

· **onStop() -** Fragment is going to be stopped by calling onStop(). Fragment becomes invisible.

· **onDestroyView() -** Destroys the view of the fragment.

· **onDestroy() -** onDestroy() is called to do final clean up of the fragment's state but Not guaranteed to be called by the Android platform.

Coding Ninjas

Handling Configuration Changes:

Some device configurations can change during runtime (such as screen orientation, keyboard availability, and when the user enables multi-window mode). When such a change occurs, Android restarts the running Activity (onDestroy() is called, followed by onCreate()). The restart behavior is designed to help your application adapt to new configurations by automatically reloading your application with alternative resources that match the new device configuration.

To properly handle a restart, it is important that your activity restores its previous state. You can use a combination of onSaveInstanceState(), ViewModel objects, and persistent storage to save and restore the UI state of your activity across configuration changes.

As a developer you might encounter scenarios where, restarting your application and restoring significant amounts of data can be costly and create a poor user experience. In such a situation, you have two other options:

Retain an object during a configuration change

Allow your activity to restart when a configuration changes, but carry a stateful object to the new instance of your activity.

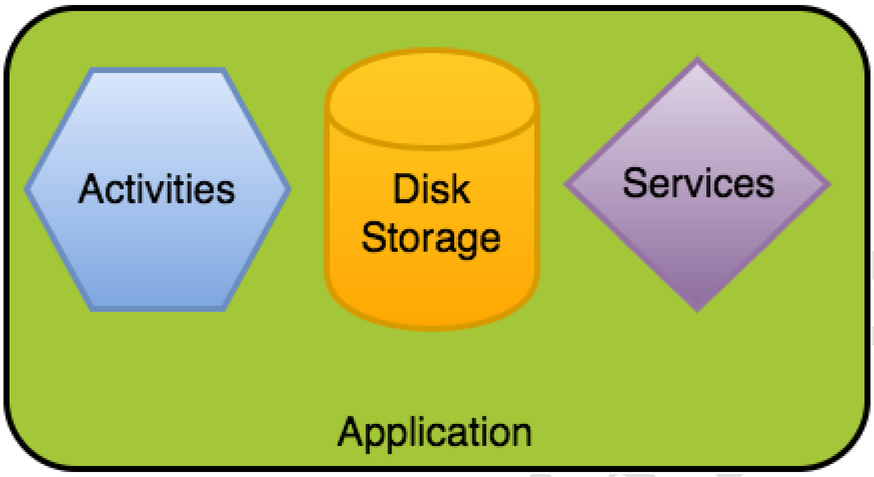
Handle the configuration change yourself

It is not recommended to handle configuration changes yourself due to the hidden complexity of handling the configuration changes. However, if you are unable to preserve your UI state using the preferred options (onSaveInstanceState(), ViewModels, and persistent storage) you can instead prevent the system from restarting your activity during certain configuration changes. Your app will receive a callback when the configurations do change so that you can manually update your activity as necessary.

Coding Ninjas

Understanding the Android Application Class:

The Application class in Android is the base class within an Android app that contains all other components such as activities and services. The Application class, or any subclass of the Application class, is instantiated before any other class when the process for your application/package is created.



img src: https://camo.githubusercontent.com/7b0b035eef781a18df47f03c5a1f0d80d8d137e0/68747470733a2f2f692e696d6775722e636f6d2f623459694166792e706e67

This class is primarily used for initialization of global state before the first Activity is displayed. Note that custom Application objects should be used carefully and are often not needed at all.

All the lifecycle methods are covered extensively in the lectures. To read more about the methods and callbacks, refer to official documentation:

1) **For Activity Lifecycle**:

   https://developer.android.com/guide/components/activities/activity-lifecycle

2) **For fragments lifecycle:**

   https://developer.android.com/guide/components/fragments

3) **Timber Library:**

   https://github.com/JakeWharton/timber

4) **Application Class:**

https://github.com/codepath/android_guides/wiki/Understanding-the-Android-Application- Class

Coding Ninjas