Android Lifecycle

Single Activity:

I scenario: when the back button is pressed -> the state is not saved and onDestroy() is called; whenever we open the app again the Bundle in onCreate() will be null

II scenario: when user navigates away (such as pressing the home button; switching to another app, accepting a call, etc.) -> the system uses onSaveInstanceState to save the app state in case the system kills the process later on; assuming that the process isn’t killed the instance is kept in the memory, holding the state and when the activity comes back to the foreground it doesn’t have to reinitialize the components to recall this information;

III scenario: a configuration change like rotation -> users should continue where they left; the activity is completely destroyed and the recreated again but the state is saved and restored for the new instance; the Bundle in onCreate() and onRestoreInstanceState() are the same

IV scenario: the app is paused by the system such as another app partially covers the running app – a purchase dialog, a third-party login dialog etc. -> the app is onPause() method;

If the dialogs are in the same app (AlertDialog or DialogFragment) this won’t cause the app to pause. Also if the user receives a notification or pulls down the notification bar.

Multiple Activities:

I scenario: if a second activity is started the first activity is Stopped (but not destroyed)

When the back button is pressed in second activity to go to first activity the second activity is being destroyed

II scenario: configuration change -> all the activities in the stack needs to restore their state after a configuration change

We should always be prepared to restore the state in any situation because the system can kill our app at any time.

III scenario: the OS kills the app

Fragments:

I scenario: Activity with Fragment -> the onCreate() of an Activity is guaranteed to start before the Fragment’s. However, the others methods are parallel and there is no order which method will be called Fragment’s or Activity’s.

II scenario: Activity with Fragment is rotated -> Fragment state is saved and restored in very similar way to activity state. The difference is that there is no onRestoreInstanceState in fragments but the Bundle is available in fragment’s onCreate, onCreateView and onViewCreated; Fragments can also be retained, which means that the same instance is used on configuration change;

III scenario: Activity with retained Fragment is rotated -> the fragment is not destroyed nor created after the rotation because the same fragment instance is used after the activity is recreated; using retained fragments is not recommended unless they are used to store data across configuration changes

ViewModels:

The lifecycle of a ViewModel is quite simple; they have only one callback: onCleared(); the initialization happens whenever you obtain the ViewModel which is normally done in onCreate()

Back Stack:

A task’s back stack is tied together with the back button, but it goes both ways. When you start a new activity using startActivity() that is pushing the new activity on top of the previous activity causing it to pause. If we press the back button the second activity gets destroyed and removed from the back stack and we are being taken back to the previous activity.

The back stack also applies to Fragments. We can use addToBackStack() to effectively add the FragmentTransaction to the back stack