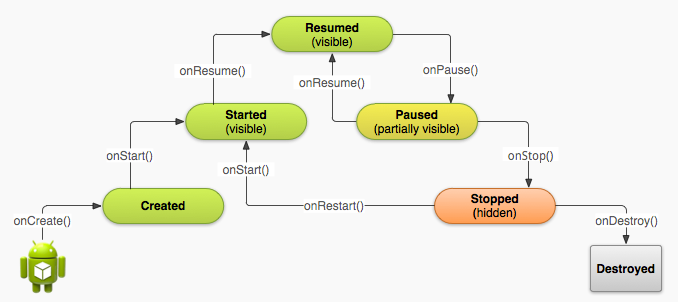
Android Exercises

Activity Life Cycle and Intent

* Activities are a fundamental building block of Android applications and they can exist in a number of different states.
* The activity lifecycle begins with instantiation and ends with destruction, and includes many states in between.
* When an activity changes state, the appropriate lifecycle event method is called, notifying the activity of the impending state change and allowing it to execute code in order to adapt to that change.



**onCreate**: called to set up the Java class for the instance of the app

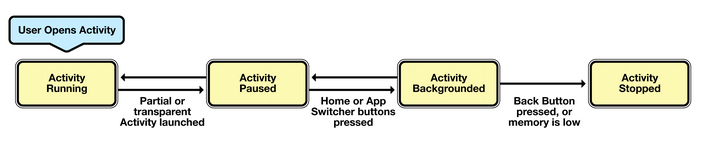
**onStart**: technically, called to initiate the “visible” lifespan of the app; at any time between onStart and onStop, the app may be visible. We can either be onResume’d or onStop’ped from this state. Note that there is also an event for onRestart, which is called before onStart if the application is transitioning from onStop to onStart instead of being started from scratch.

**onResume**: technically, the start of the “foreground” lifespan of the app, but this does not mean that the app is fully visible and should be rendering – more on that later

**onPause**: the app is losing its foregrounded state; this is normally an indication that something is fully covering the app. On versions of Android before Honeycomb, once we returned from this callback, we could be killed at any time with no further app code called. We can either be onResume’d or onStop’ped from this state

**onStop**: the end of the current visible lifespan of the app – we may transition to on(Re)Start to become visible again, or to onDestroy if we are shutting down entirely. Once we return from this callback, we can be killed at any time with no further app code called on any version of Android.

**onDestroy**: called when the Java class is about to be destroyed. Once this function is called, there is only one option for transition (other than being killed): onCreate.

**Activity States:** 

**Active or Running** - Activities are considered active or running if they are in the foreground, also known as the top of the activity stack. This is considered the highest priority activity in Android, and as such will only be killed by the OS in extreme situations, such as if the activity tries to use more memory than is available on the device as this could cause the UI to become unresponsive.

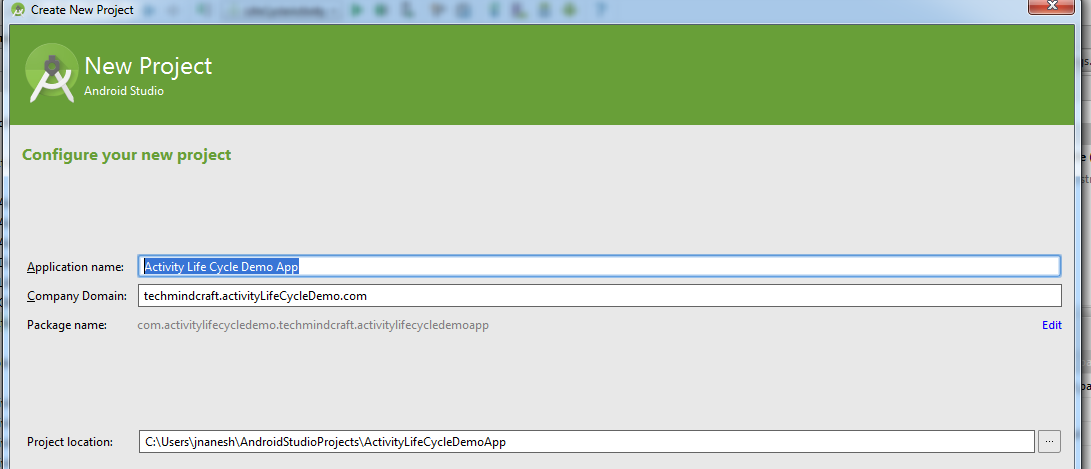
**Paused** - When the device goes to sleep, or an activity is still visible but partially hidden by a new, non-full-sized or transparent activity, the activity is considered paused. Paused activities are still alive, that is, they maintain all state and member information, and remain attached to the window manager. This is considered to be the second highest priority activity in Android and, as such, will only be killed by the OS if killing this activity will satisfy the resource requirements needed to keep the Active/Running Activity stable and responsive.

**Stopped/Backgrounded** - Activities that are completely obscured by another activity are considered stopped or in the background. Stopped activities still try to retain their state and member information for as long as possible, but stopped activities are considered to be the lowest priority of the three states and, as such, the OS will kill activities in this state first to satisfy the resource requirements of higher priority activities.

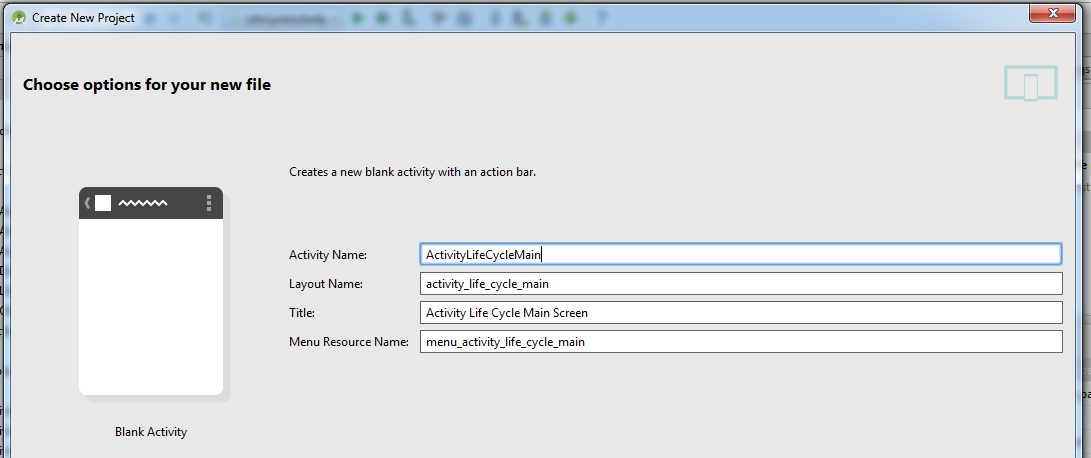
**Restarted** – It is possible for an activity that is anywhere from paused to stopped in the lifecycle to be removed from memory by Android. If the user navigates back to the activity it must be restarted, restored to its previously saved state, and then displayed to the user.

**Exercise on Activity Life Cycle and Intents in Android:**

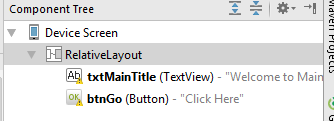
1. Create a new project and name it as “Activity Life Cycle Demo App”



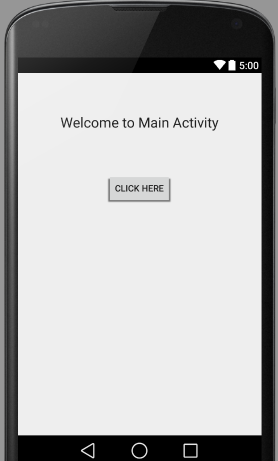
1. Enter the Activity Name as “ActivityLifeCycleMain”, Layout Name as “activity\_life\_cycle\_main” and Title as “Activity Life Cycle Main Screen”

“

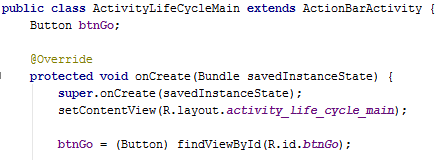
1. In the “activity\_life\_cycle\_main.xml” file add a Text View and Button and name them as below:



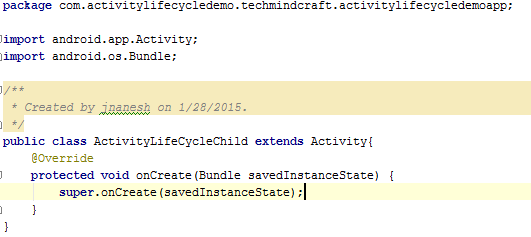
1. Your “activity\_life\_cycle\_main.xml” should look like as below:



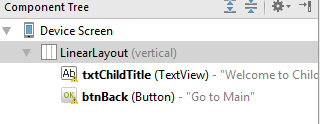
1. Declare the Button and set reference in your class file as below:



1. Create an another activity and name it as “ActivityLifeCycleChild”



1. Create a Layout file and name it as “activity\_life\_cycle\_child” and add TextView and Button as mentioned below:



1. Your “activity\_life\_cycle\_child.xml” should look like as below:

