UNIT 3 ACTIVITY LIFECYCLE

3.0 INTRODUCTION

In this unit you will be exploring the Activity Lifecycle of an Android application. After an introduction to *Activity*, *Activity life cycle* and *life cycle methods* are discussed. By watching the screen cast developed for this particular unit, you will get a better understanding of how activity lifecycle works. The demonstration on how to determine the inter-process dependencies at runtime will also be further explained in the screencast.

3.1 OBJECTIVES

Outcomes

- Sketch the activity life cycle diagram and identify the components.
- List the status of the activity life cycle and describe each status related to the working mobile application.
- Explain the process of the activity life cycle, foreground, visible, background and empty processes.

After studying this unit, you should be able to:

Terminology

activity:	Activity is an application component that provides a screen with which users can interact
life cycle:	journey of an Activity passing through different stages of life

3.2 WHAT IS AN ACTIVITY IN ANDROID?

An Activity in Android is an application component that provides a screen with which users can interact in order to do something, such as dial the phone, take a photo, send an email, or view a map as explained in the previous unit. Each activity is given a window in which to draw its user interface. The window typically fills the screen, but may be smaller than the screen and float on top of other windows.

When you open an application, the very first screen that appears in front of you, is called a default Activity (or Main Activity). There can be more than one loosely coupled Activities in your application. Generally, as the complexity of an application increases, the need of number of Activities increases proportionally.

3.2.1 Check Your Progress

Give an example of an Activity of an Android application

Now you have an idea about what an Activity is. Let us look into more details about how Activities are interrelated. An Activity can start another Activity to perform some actions. If it does so then system stops the current Activity and preserves it in a stack called activity stack which will be discussed in the next section.

Activity Stack

Typically, one activity in an application is specified as the "main" activity, which is presented to the user when launching the application for the first time. Each activity can then start another activity in order to perform different actions. Each time a new activity starts, the previous activity is stopped, but the system preserves the activity in a stack called activity stack or "back stack". It is illustrated in the Figure 3.1.

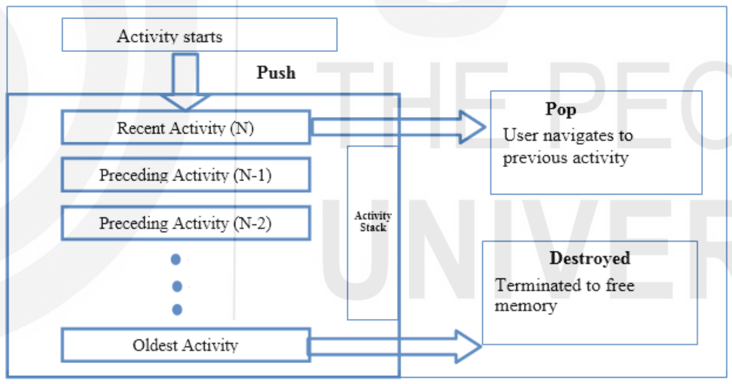

Figure 3.1: Activity Stack

When a new activity starts, it is pushed into the back stack and takes user focus. The back stack abides to the basic "last in, first out" stack mechanism, so that when the user is done with the current activity and presses the Back button, it is popped from the stack and the previous activity resumes. The old activities are destroyed by the OS to free up memory. But activity stack keeps activity reference and re-launch an activity when needed.

3.2.2 Check Your Progress

Draw and briefly explain, how each new activity in a task adds an item to the back stack by referring to the following link. https://developer.android.com/guide/components/tasks-and-back-stack.html

Hint: You can consider three activities at a time and progress between them

Once an activity is launched, it goes through a lifecycle called Activity Lifecycle; a term that refers to the steps the activity progresses through as the user (and operating system) interacts with it.

3.3 WHAT IS AN ACTIVITY LIFECYCLE?

From its creation to its conclusion, an Activity goes through many stages of its life such as start, pause, resume, stop, etc. This journey of Activity passing through different stages of life called lifecycle of the Activity. Every stage of the Activity lifecycle has a specific method associated to it, called call-back method. When an Activity stops and another starts, it means a call-back method is called for each Activity. The lifecycle of an Activity is managed by the Android run time system.

Generally, an Activity can remain in following four states:

- If an activity is in the foreground of the screen (at the top of the stack), it is *active* or *running*.

If an activity has lost focus but is still visible (that is, a new non-full-sized or transparent activity has focus on top of your activity), it is paused. A paused activity is completely alive (it maintains all state and member information and remains attached to the window manager), but can be killed by the system in extreme low memory situations.

- If an activity is completely obscured by another activity, it is *stopped*. It still retains all state and member information, however, it is no longer visible to the user so its window is hidden and it will often be killed by the system when memory is needed elsewhere.

- If an activity is paused or stopped, the system can drop the activity from memory by either asking it to finish, or simply killing its process. When it is displayed again to the user, it must be completely restarted and restored to its previous state.

- Figures 3.2 depicts the state diagrams to represent the different stages of the Activity lifecycle with different call-back methods.

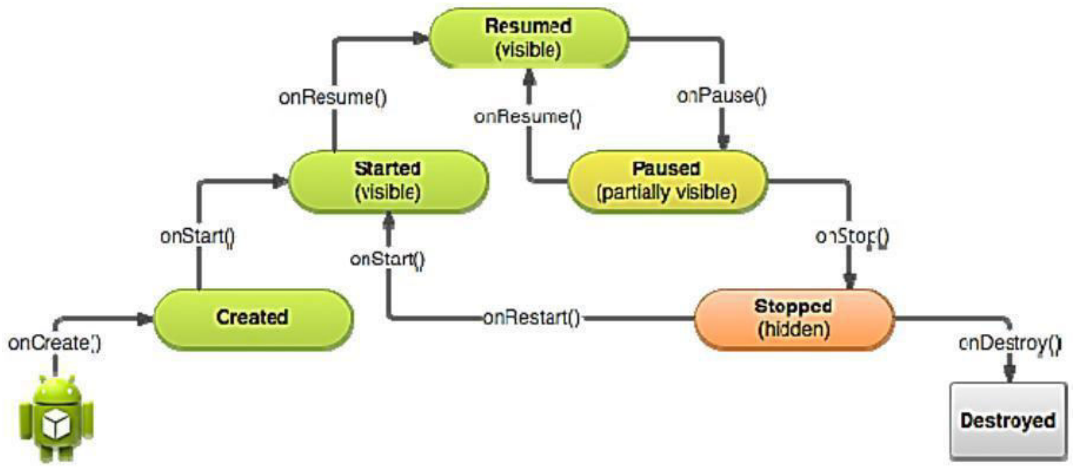

Figure 3.2: A simplified illustration of Activity Lifecycle

(Source: http://developer.android.com/training/basics/activity-lifecycle/starting.html)

3.3.1 Video-V4: Android Application Fundamentals

Let us watch this video and understand the android activity life cycle that moves to each state and respond to different call-back methods. This video will give you a detailed explanation of an Activity, Activity lifecycle, use of back-stack and the different call-back methods of Activity Lifecycle.

URL:https://tinyurl.com/y7czgm5f

Figure 3.3 illustrates the loops and the paths of an activity that might take place between states.

The rectangles represent the call-back methods you can implement to perform operations when the activity transitions between states. You can see that when a process is killed and the user navigates back to onCreate() method.

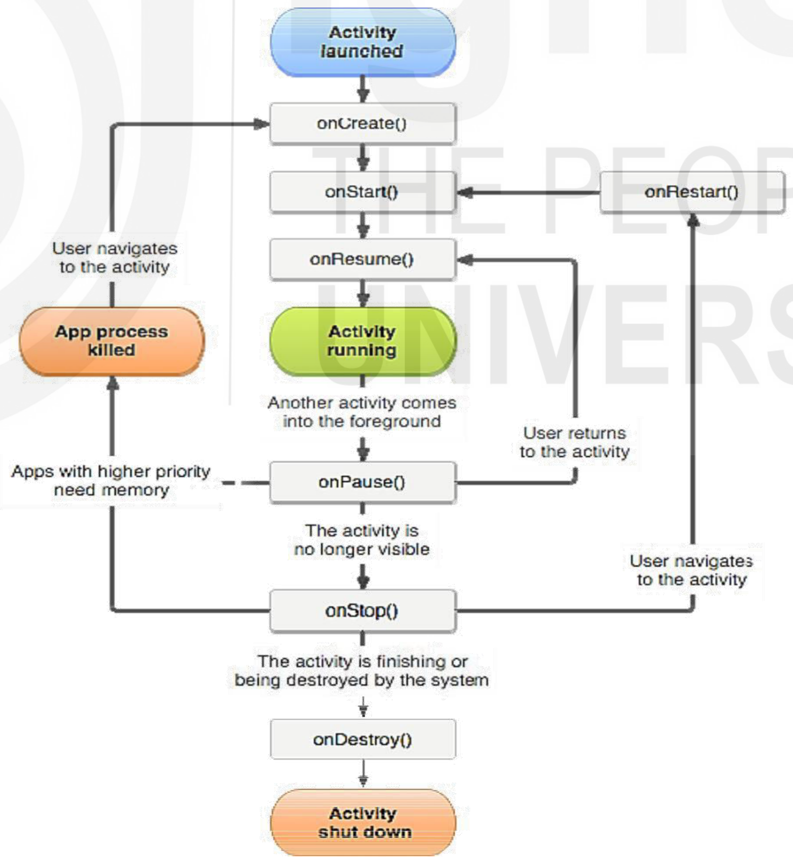
Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to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

Figure 3.3: Activity Lifecycle
(Source:http://developer.android.com/guide/components/activities.html)

Now you have learnt the Activity Lifecycle and its states. Next, we are going to discuss how these states will be implemented as methods.

Activity Lifecycle Methods

The Android system defines a lifecycle for activities via predefined lifecycle methods. The following Table 3.1 is showing each lifecycle call-back method with details such as name of the method, description of the method, what method is called after the specified method, method is killable or not, etc.

Table 3.1: A summary of the activity lifecycle's call-back methods

Methods	**Description**	**Killable after?**	**Next**
onCreate()	Called when the activity is first created. This is where you should do all of your normal static set up such as create views, bind data to lists, and so on.	No	onStart()
onRestart()	Called after the activity has been stopped, just prior to it being started again.	No	onStart()
onStart()	Called just before the activity becomes visible to the user.	No	onResume() or onStop()
onResume()	Called just before the activity starts interacting with the user. At this point the activity is at the top of the activity stack.	No	onPause()
onPause()	Called when the system is about to start resuming another activity.	Yes	onResume() or onStop()
onStop()	Called when the activity is no longer visible to the user. This may happen because it is being destroyed, or because another activity (either an existing one or a new one) has been resumed and is covering it.	Yes	onStart() or onDestroy()
onDestroy()	Called before the activity is destroyed. This is the final call that the activity will receive	Yes	Nothing

(Source:http://developer.android.com/guide/components/activities.html)

Within the lifecycle call-back 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.

To get the detailed information of working of an Activity, pursue the following link:

http://developer.android.com/guide/components/activities.html

With the understating of the Activities, Activity Lifecycle and call-back methods, we can discuss how these Activities are managed in the Android application.

Managing Activities in the application

All Android applications started by the user are remained in memory, which makes restarting applications faster. But in reality the available memory on an Android device is limited. To manage these limited resources the Android system is allowed to terminate running processes or recycling Android components.

As stated earlier Android applications run within instances of the Dalvik virtual machine with each virtual machine being viewed by the operating system as a separate process. If the system identifies that resources on the device are reaching capacity it will take steps to terminate processes to free up its memory.

If the Android system needs to free up resources it follows logic. Every process gets a priority. If the Android system needs to terminate processes it follows the priority system. You will learn the Android process states and priority system in next section.

3.4 WHAT ARE THE ANDROID PROCESS STATES?

When deciding as to which process to terminate in order to free up memory, the system consider both the priority and state of all currently running processes. It is considered by Google as an important hierarchy. Processes are then terminated starting with the lowest priority and working up the hierarchy until sufficient resources have been liberated for the system to function. As outlined in Figure 3.4, a process can be in one of the following five states at any given time of priority.

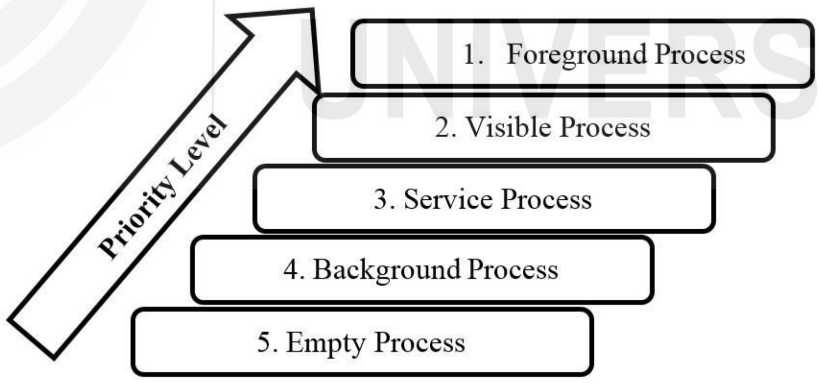

Figure: 3.4: Android process states and priority levels

Let us explain the priority levels more in details below.

Foreground Process

These processes are assigned the highest level of priority. It is one that is required for what the user is currently doing. Various application components can cause its containing process to be considered foreground in different ways. A process is considered to be in the foreground if any of the following conditions hold:

- Hosts an activity with which the user is currently interacting.
- Hosts a Service connected to the activity with which the user is interacting.
- Hosts a Service that has indicated, via a call to startForeground(), that termination would be disruptive to the user experience.
- Hosts a Service executing either its onCreate(), onResume() or onStart() callbacks.
- Hosts a Broadcast Receiver that is currently executing its onReceive() method.

Visible Process

It is a process containing an activity that is visible to the user but is not the activity with which the user is interacting. Such a process is considered extremely important and will not be killed unless doing so is required to keep all foreground processes running. This may occur, for example, if the foreground Activity is displayed as a dialog that allows the previous Activity to be seen behind it.

Service Process

A process that contain a Service that has already been started and is currently executing is called a service process. However, these processes are not directly visible to the user. These processes are generally doing things that the user cares about such as background mp3 playback or background network data upload or download. The system will always keep such processes running unless there is not enough memory to retain all foreground and visible processes.

Background Process

It is a process that contains one or more activities that are not currently visible to the user, and does not host a Service that qualifies for Service Process status. These processes have no direct impact on the user experience. Provided they implement their Activity life-cycle correctly, the system can kill such processes at any time to reclaim memory for one of the three previous processes types.

Empty Process

Empty processes no longer contain any active application component. They are held in memory ready to serve as hosts for newly launched applications. Such processes are, obviously, considered to be the lowest priority and are the first to be killed to free up resources.

3.4.1 Check Your Progress

Select True/False statements
You can select A) True or B) False for the following statements

1. There is no guarantee that an activity will be stopped prior to being destroyed.
2. During an activity lifecycle, onStart() is the first callback method invoked by the system.
3. Finish() method is used to close an activity.
4. Once the onStop() method is called, the activity is no longer visible.
5. When onPause() method is called in an activity, another activity gets into the foreground state.

Hint check your answers with Answer guide at the end of this unit.

Now you have learnt how activities are managed. The situation of deciding the highest priority process is sometimes complex than outlined in the previous section since that processes can often be inter-dependent. Inter-Process Dependencies are explained further in the following section.

Inter-Process Dependencies

Android system will also take into consideration that whether the process is in some way serving another process of higher priority. Likewise, when deciding as to the priority of a process the system consider inter-dependencies.

For instance, consider a service process acting as the content provider for a foreground process. The Android documentation states that a process can never be ranked lower than another process if it is currently serving a foreground process. Thus, the systems manage the activities and memory facilitating fast running applications.

(Source: https://developer.android.com/training/articles/memory.html)

3.5 SUMMARY

Mobile applications are common in day to day life. For instance, we can recall some of them as Calculator app, Date and activity schedule app, Map finding etc. All these application components provides a screen for user to interact and perform something with the mobile device. Those components are made of multiple activities providing various user interfaces. In this unit we described what an activity is and the activity lifecycle in a mobile application.

Activity lifecycle has few states and those states depend on the process states of it changes based on pre-defined priority levels. Moreover, activity lifecycle has methods which can be called upon wherever necessary. The rest of unit discussed the basics of how to build and use an activity.

3.6 FURTHER READINGS

- https://developer.android.com/guide/components/activities/intro-activities
- https://developer.android.com/guide/components/activities/activity-lifecycle
- https://learncswithandroid.blogspot.com/2017/12/android-process-states.html