MAJLIS ARTS AND SCIENCE COLLEGE

PG DEPARTMENT OF COMPUTER SCIENCE

(Affiliated to the University of Calicut, approved by the Government of Kerala)

Majlis Nagar, Puramannur-P.O 676552, Malappuram Dt, Kerala, 0494 2643970, 9539111174

MAJLIS EDUCATIONAL COMPLEX
AMLI
Estd. 1988
PURAMANNUR, KERALA

DEPARTMENT OF COMPUTER SCIENCE • SINCE 1995
MAJLIS COLLEGE

Sixth Semester Online Study Camp.

Study Camp. 6 BCA , BSc

WhatsApp group

Scan QR Code to Join Study Camp WhatsApp Group

*** UNIT WISE REVISION**
*** PREVIOUS YEAR QUESTION PAPER DISCUSSION**
*** ASSIGNMENTS**

masc.majliscomplex.org

ANDROID PROGRAMMING – MODULE - 1

1. History of Android

The story of Android dates back to 2003 when Andy Rubin, Rich Miner, Nick Sears, and Chris White co-founded a start-up Android Inc. in Palo Alto, California. However, the company was later faced with the insufficiency of funds which brought Google into the picture. Google could sense the potential the product carried within and sealed a deal worth $50 Million to acquire Android in 2005.

2. Android Version Series

Android 1.0 (API 1) and 1.1 (API 2)

Android 1.0 (API 1) was launched on the 23rd Of September 2008. It was incorporated into the HTC Dream smartphone (aka T-mobile G1 in the US). It thus became the first-ever Android device. The features it offered included Google Maps, YouTube, an HTML browser, Gmail, camera, Bluetooth, Wi-Fi, and many more. The unique feature at that time was the presence of an Android Market (now Play Store) from where the users could download and update Android applications additional to what was already pre-installed.

- Saving of attachments in messages
- Availability of details and reviews for businesses on Google Maps
- Longer in-call screen timeout by default while the speakerphone was in use along with the ability to toggle the dial-pad.
- Support was added for a marquee in the system layouts.

Android 1.5 (API 3) aka Cupcake

This version came up in late April 2009 and was the first to have Google's dessert-themed naming scheme and be incorporated in the Samsung Galaxy phone series. It was introduced with a lot of functionalities that we take for granted today. These updates included new features and enhancements to the ones already present in the above versions, for example, some major updates included auto-rotation, third-party keyboard support, support for widgets, video recording, enabling copy-paste for browser, facility to upload videos on YouTube, check phone usage history, etc.

Android 1.6 (API 4) aka Donut

Just after a couple of months in September 2009, Donut was released. One of its most significant features was the inclusion of CDMA-based networks that made it possible for carriers across the globe to support it. Earlier only GSM technologies were in use. Other critical improvements such as support for devices having different screen sizes, quick search boxes and bookmarks on web browsers, fuller integration of Gallery, Camera, and Camcorder with faster camera access, expansion of the Gesture framework, text-to-speech, etc. were too introduced in this update.

Android 2.0 (API 5), 2.0.1 (API 6), and 2.1 (API 7) aka Éclair

Almost after a year of the Version 1 release, Version 2.0 was launched in October 2009. Key highlights of this update included the introduction of navigation in Google Maps with voice guidance, support for adding multiple accounts in one device, display of live wallpapers, a lock screen with drag and drop unlocking functionality, additions to camera services such as Flash and digital zoom, the inclusion of smarter dictionary for virtual keyboards

Android 2.2 (API 8) aka Froyo.

Froyo is actually a combination of the words, “frozen Yogurt”. This version was launched in May 2010. Some of its most significant features included Wi-Fi mobile hotspot support, push notifications through Android Cloud to Device Messaging, enhancement of device security through PIN/ Password protection, Adobe Flash support, USB tethering functionality, update in Android Market application with the automatic update of apps features, support for Bluetooth enabled car, etc.

Android 2.3 (API 9) aka Gingerbread

Gingerbread, released even before the later versions of Froyo, brought drastic changes to the look and feel of smartphones. The first phone to adopt this version was Nexus S, co-developed by Google and Samsung. In this version, the user interface design was updated to bring in more simplicity and speed. Support for extra-large screen sizes and resolutions was integrated.

Android 3.0 (API 11) Honeycomb

In Feb 2011, Android 3.0 Honeycomb was released to be installed on tablets and phones with larger screens only and had functions that could not be managed on phones with smaller screens. The most important function brought by this version was to eliminate the need for the physical button and rather the introduction of virtual buttons for performing the start, back, and menu functions. This version was first launched along with the Motorola Xoom Tablet.

Android 4.0 (API 14) Icecream Sandwich

Android 4.0 was released in October 2011. It was a combination of a lot of features of the Honeycomb Version and the Gingerbread. For the first time ever, the Face Unlock feature for smartphones was introduced with 4.0. Other prominent features included the possibility to monitor the use of mobile data and Wi-Fi, sliding gestures to reject notifications, tabs of a browser or even tasks, integration of screenshot capture using the Power and Volume button, real-time speech to text dictation, using certain apps without necessarily unlocking, pre-fed text responses to calls, and many more.

Android 4.4 (API 19) Kitkat

The launch of Android 4.4 KitKat took place in 2013 and had many distinct features such as the blue accents found in Ice Cream Sandwich and Jelly Bean had turned whiter and many storage applications displayed lighter colour schemes. With the ‘Ok Google’ command, a user could access Google at any time and could work on phones with a minimum RAM memory of 512MB.

Android 5.0 (API 21) Lollipop

Android 5.0 was launched in Nov 2014 with the Nexus 6 device. It was the first to feature Google's 'Material Design' Philosophy which brought tremendous improvements to the UI Design, for example, the Vector drawable which could be scaled indefinitely without losing their definition were introduced. Other significant features included the replacement of VM Dalvik with Android Runtime that improved the app performance and responsiveness considerably as some of the processing power for applications could now be provided before they were opened

Android 7.0 (API 24) Nougat

In August 2016, Google released Android 7.0 Nougat. It presented improved multitasking features especially for devices with larger screens, for example, the spilt-screen mode was introduced along with provision for fast switching between applications. Other significant features included the integration of the Daydream virtual-reality platform and enhancement of the 'Doze now 'mode.

Android 8.0 (API 26) Oreo

This version appeared in Aug 2017 and brought a series of noteworthy changes to the existing ones. It turned out to be a more powerful and faster version as it had a 2x boot speed compared to Nougat when tested on Pixel devices, according to a claim by Google. This version was smarter as well which was evident through the introduction of functionalities such as Autofill, picture-in-picture mode (for example, the video calling window in WhatsApp while working with some other app), and the Notification dots through which user could quickly catch up with newer information.

Android Version 9 Pie (API 28)

Android 9 was introduced in August 2018. It brought tremendous improvements to the visual aspect and made exceptional use of the power of artificial intelligence. The most noticeable ones included, replacement of traditional navigation buttons with an elongated button in the centre that functioned as the new start button, swiping which up provided an overview of recently used applications, a search bar, and five application suggestions.

Android Version 10 (API 29)

With Android 10 in Sept 2019, Google announced a rebranding of the operating system, eliminating the sweets-name based naming scheme that was being used for the earlier versions. With this version, a new logo and a different colour scheme were announced. Facilities such as Live Captions for all media, smarter replies to text (automated text and actions suggestions), 'Focus mode' to block out distractions by selecting certain apps to pause temporarily, replacement of navigation buttons with the use of gestures, availability of the dark mode at the system level, provision for more control over

Android 11

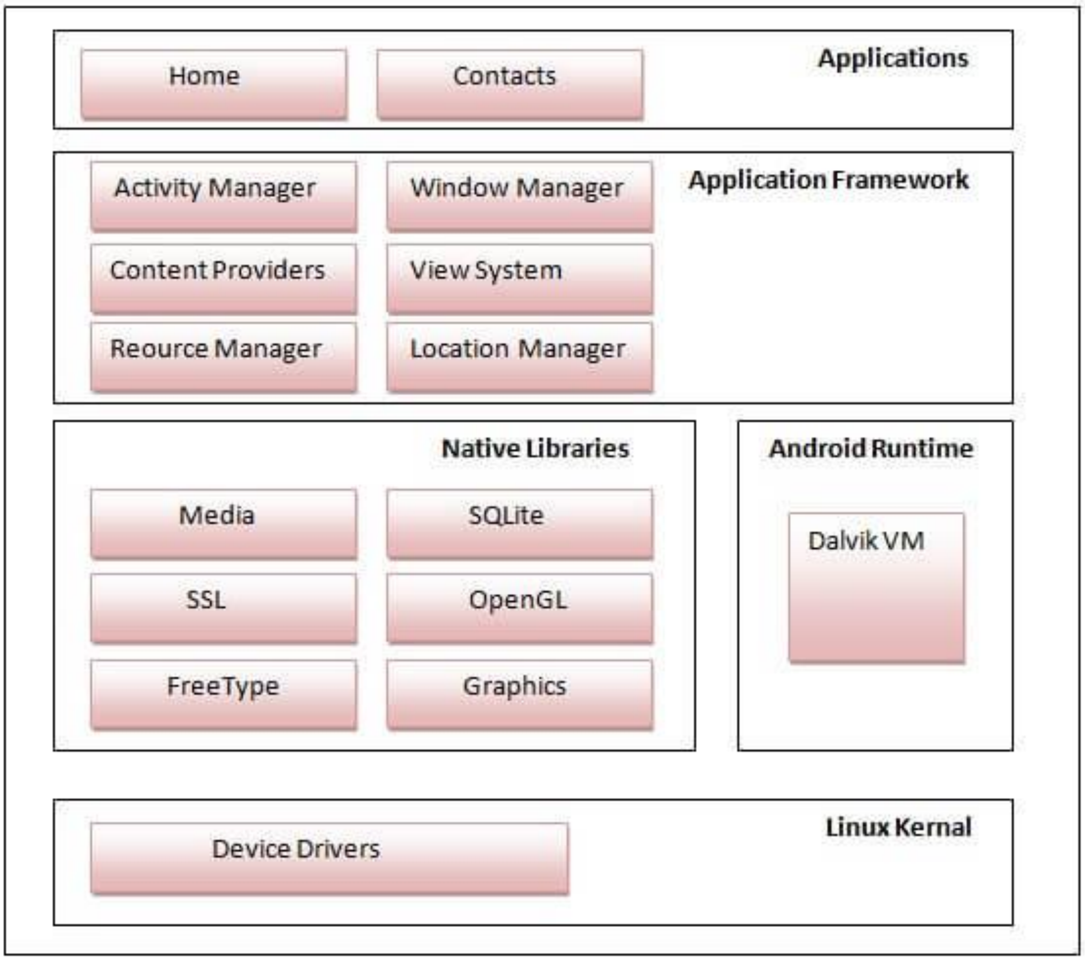
Android Version Android 11 recently released on the 8th of September 2020. This version has come up with a tagline 'The OS that gets to what's important 'and it's pretty much justified. Android 11 brings along capabilities to control conversations across multiple messaging apps all in the same spot, it allows the user to digitally select priorities for people they are conversing with and then show the most important conversations at the top and on the lock screen.11 (API 30)

3. Android architecture or Android software stack

android architecture or Android software stack is categorized into five parts:

- Linux kernel
- native libraries (middleware),
- Android Runtime
- Application Framework
- Applications

Let's see the android architecture first.



1) Linux kernel

It is the heart of android architecture that exists at the root of android architecture. Linux kernel is responsible for device drivers, power management, memory management, device management and resource access.

2) Native Libraries

On the top of linux kernel, there are Native libraries such as WebKit, OpenGL, FreeType, SQLite, Media, C runtime library (libc) etc. The WebKit library is responsible for browser support, SQLite is for database, FreeType for font support, Media for playing and recording audio and video formats.

3) Android Runtime

In android runtime, there are core libraries and DVM (Dalvik Virtual Machine) which is responsible to run android application. DVM is like JVM but it is optimized for mobile devices. It consumes less memory and provides fast performance.

4) Android Framework

On the top of Native libraries and android runtime, there is android framework. Android framework includes Android API's such as UI (User Interface), telephony, resources, locations, Content Providers (data) and package managers. It provides a lot of classes and interfaces for android application development.

5) Applications

On the top of android framework, there are applications. All applications such as home, contact, settings, games, browsers are using android framework that uses android runtime and libraries. Android runtime and native libraries are using Linux kernel.

4. Android SDK

Android SDK is a software development kit developed by Google for the Android platform. The Android SDK allows you to create Android apps, and you don't need to be an expert to use it. In this tutorial, I'll explain what the Android SDK is and how to get started with it.

Android SDK comes bundled with Android Studio, Google's official integrated development environment (IDE) for the Android operating system. You can learn about Android Studio and the Android App Development Kit in another of my articles.

The Android SDK is a collection of software development tools and libraries required to develop Android applications. Every time Google releases a new version of Android or an update, a corresponding SDK is also released which developers must download and install. It is worth noting that you can also download and use the Android SDK independently of Android Studio, but typically you'll be working through Android Studio for any Android development.

The Android SDK comprises all the tools necessary to code programs from scratch and even test them. These tools provide a smooth flow of the development process from developing and debugging, through to packaging.

5. Android Project File Structure

Packages in Java are essentially just folders where your classes can be stored. This allows you to write code that has a well-organized structure where classes that are related can be in the same package and be named in a meaningful way that indicates their purpose.

You've already used packages before in Java, perhaps without realizing it, because it's almost impossible to write a meaningful application that doesn't use classes in different packages. For example, Java defines a top-level package named java, and inside it is packages such as lang, which contains core classes like String, and util, which contains collection classes like ArrayList. Similarly, the Android API provides a top-level android package that contains packages named graphics, view, widget, and many more.

src/

Contains source code that you write for your app.

gen/

Contains source code that is autogenerated by the Android development tools that is required for your app. Do not modify any of the source files in this folder—your changes will be overwritten the next time you build your project anyway.

libs/

Contains precompiled third-party libraries (JAR archives) that you want to use in your app. For example, if you were writing an app that pulls data from Twitter feeds, you would want to use a Twitter library that someone has already written for you, and you would put it in this folder.

res/

Contains other folders with resources for your application: GUI layouts, icons, menus, and so forth.

assets/

Contains other media that you might want to use in your app, such as videos, sounds, and large images that are not used directly in GUI layouts.

6. Android Virtual Device

An Android Virtual Device (AVD) is an emulator configuration that allows developers to test the application by simulating the real device capabilities. We can configure the AVD by specifying the hardware and software options. AVD manager enables an easy way of creating and managing the AVD with its graphical interface. We can create as many AVDs as we need, based on the types of device we want to test for. Below are the steps to create an AVD from AVD manager graphical interface. Go to Window ->AVD Manager and select Virtual Devices. Click on New to create a Virtual Device, give it some Name and select Target Android Platform from the drop-down list Click “Create AVD” and we are done!

7. Structure of an Android App

The structure of an Android application is fairly rigidly defined. In order for things to work properly, you need to put keep certain files in the right places. At the end of this chapter, we also discuss how basic app and screen navigation works on Android devices.

Packages in Java

Packages in Java are essentially just folders where your classes can be stored. This allows you to write code that has a well-organized structure where classes that are related can be in the same package and be named in a meaningful way that indicates their purpose.

Application Resources

menu/

Contains XML files that describe menus that are associated with screens in your application, which the user can make appear by clicking the Menu button on his or her device.

drawable-*dpi/

Contains images used in various parts of the application: launch screen icons, images attached to buttons or menus, and so forth.

Android projects can actually have multiple drawable folders that are named after the resolution of the device. For example, drawable-ldpi would contain images used on low resolution devices, drawable-hdpi would contain images used on high resolution devices, and so forth. This allows an app developer to create images designed specifically for different sized screens that look crisp and clear without any scaling or blurring.

Of course, if an exact resolution match is not found, the Android OS will look for the closest match and scale it up or down to fit. For this reason, in this class we'll just stick all of our images in drawable-hdpi and not worry about multiple sizes.

layout/

Contains XML files that describe the layout of the widgets (buttons, text fields, and so forth) on the screens in your application. Fortunately, you typically do not need to write any XML directly – the Android tools in Eclipse provide a drag-and-drop editor for laying out your screens.

values/

Contains "values" used throughout the application, such as text strings and style definitions. The main use of this is the strings.xml file, which can be used to store the text strings used in your GUI layouts (e.g., the labels on buttons). You don't have to do this, but if you do, it makes your application easier to translate into other languages because you have all of the strings stored conveniently in one place and you only need to have that single file translated.

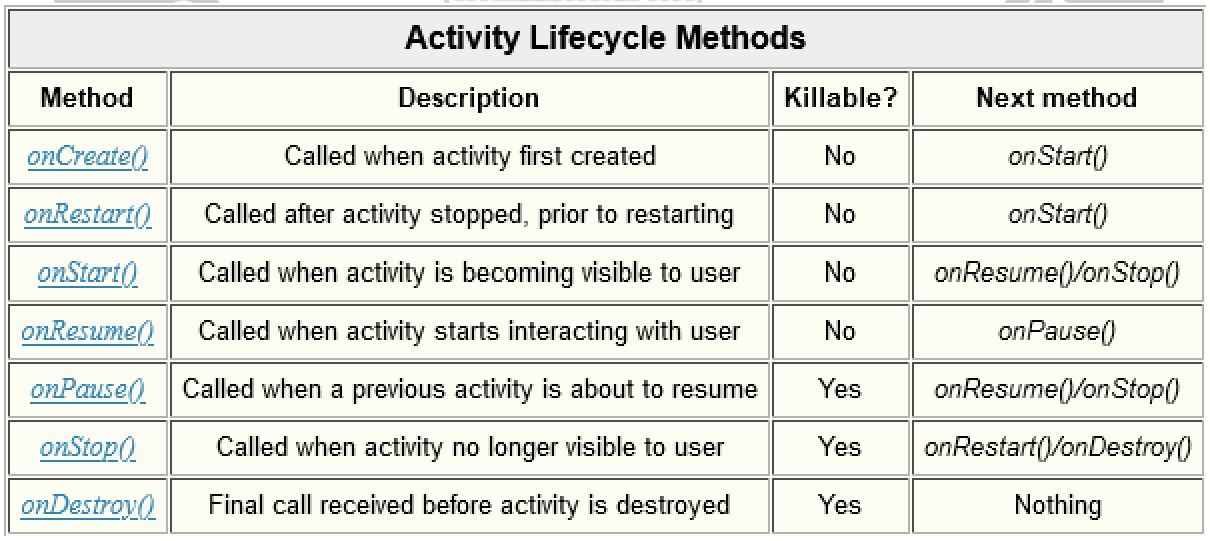
8. Processes and Application Lifecycle in Android

It is very important to have a basic understanding of the lifecycle for an activity in Android in order to efficiently manage resources on limited devices, and to ensure a seamless response for the user.

Overview of Android Lifecycles

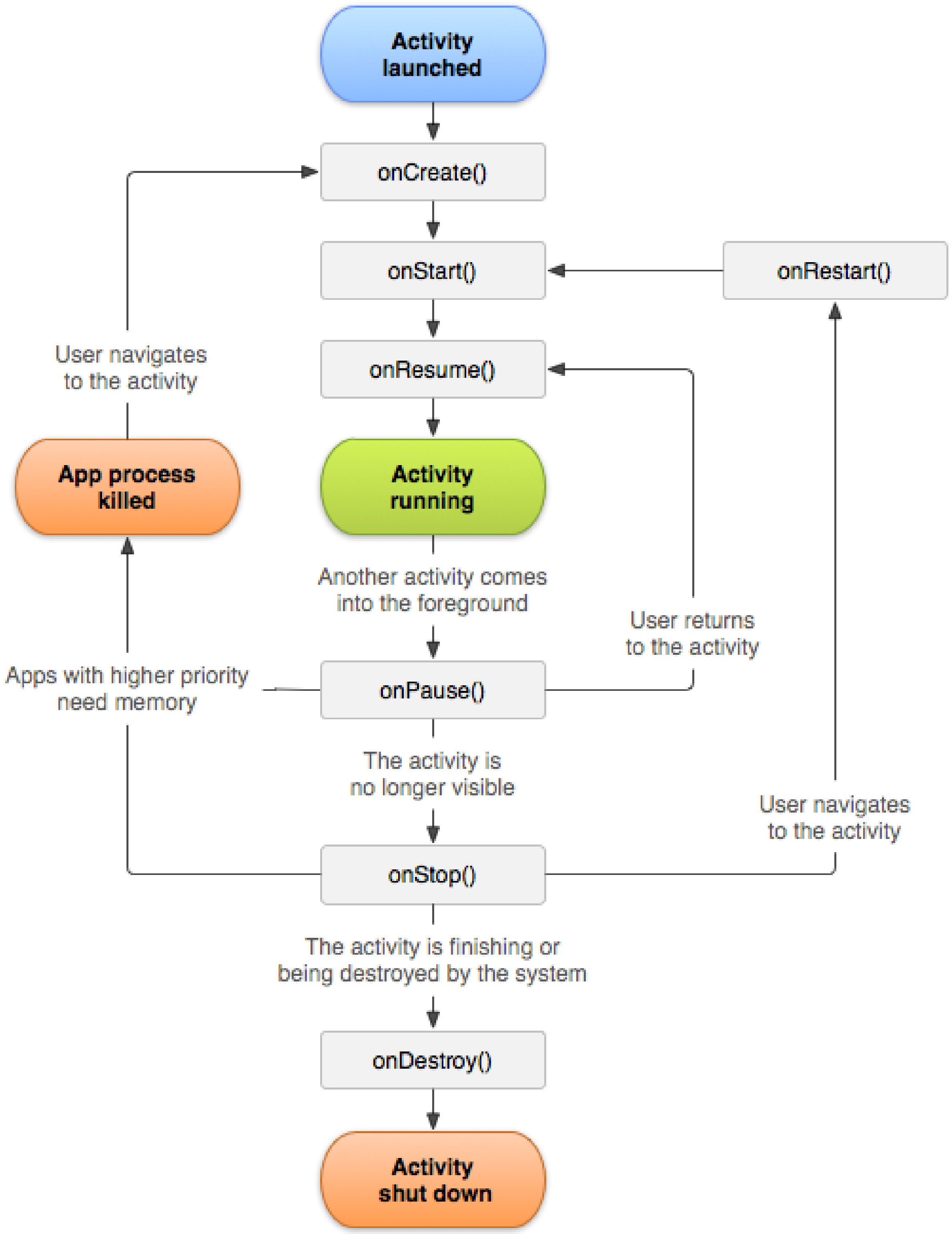
A general overview of the lifecycle of an application is shown in the following figure, which is taken from the documentation for Activity.

Notice in particular that when another activity comes to the foreground it does not generally mean that the task originally in the foreground (your app, say) is discarded. The seven methods contained in square boxes in the preceding diagram are called the lifecyle methods because they govern the lifecycles of Android applications. Their properties are summarized in the following table.



Activity Lifecycle Methods

Method	Description	Killable?	Next method
onCreate()	Called when activity first created	No	onStart()
onRestart()	Called after activity stopped, prior to restarting	No	onStart()
onStart()	Called when activity is becoming visible to user	No	onResume()/onStop()
onResume()	Called when activity starts interacting with user	No	onPause()
onPause()	Called when a previous activity is about to resume	Yes	onResume()/onStop()
onStop()	Called when activity no longer visible to user	Yes	onRestart()/onDestroy()
onDestroy()	Final call received before activity is destroyed	Yes	Nothing


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

The Three Lives of Android

It is useful to think of an application in Android having three "lifetimes" associated with the preceding diagram and table (see the documentation of Activity for a more thorough discussion).

The Entire Lifetime: the period between the first call to onCreate() to a single final call to onDestroy(). We may think of this as the time between setting up the initial global state for the app in onCreate() and the release of all resources associated with the app in onDestroy().

The Visible Lifetime: The period between a call to onStart() until a corresponding call to onStop(). Although this is termed the "visible lifetime", the app may not be directly visible and interacting with the user at any one time if it is not in the foreground. The feature that distinguishes this lifetime is that, even if not in the foreground, the app maintains resources such that it can instantaneously return to the foreground.

The Foreground Lifetime: The period between a call to onResume() until a corresponding call to onPause(). During foreground lifetime the activity is in front of all other activities and interacting with the user.