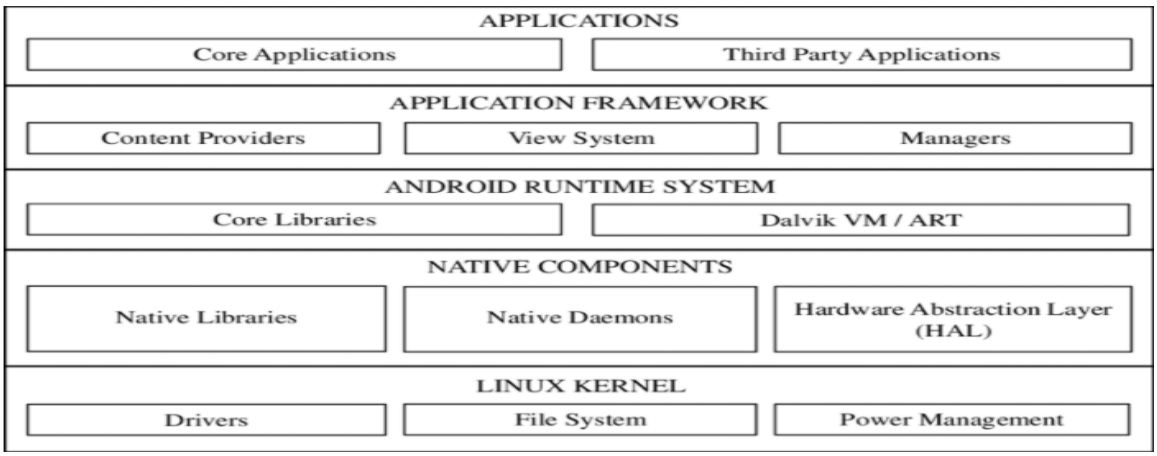
ATHARVA KALE **PRACTICAL NO. 1** **ROLL NO: 27**

AIM: To Study about Android

❖ THEORY:

A) Study android platform, the layers of android and four kinds of android components, understanding the android Manifest.xml file.

- **Android Platform:**
    1. The Android platform is a platform for mobile devices that uses a modified Linux kernel.
    2. The Android Platform was introduced by the Open Handset Alliance in November of 2007.
    3. Most applications that run on the Android platform are written in the Java programming language.
    4. Although most of the applications that run on the Android Platform are written in Java, there is no Java Virtual Machine. Instead, the Java classes are first compiled into what are known as Dalvik Executables and run on the Dalvik Virtual Machine.
    5. Android is an open development platform.
    6. To create an application for the platform, a developer requires the Android SDK, which includes tools and APIs.
- **Layers Of Android:**
    - **The following are the layers that compose the Android architecture.**



- **Application:**
    1. Android application is the top layer.
    2. All applications are installed on this layer only. Examples of such applications are Contacts Books, Browser, and Games etc.
- **Application Framework:**
    1. The Application Framework layer provides many higher-level services to applications in the form of Java classes.
    2. Application developers are allowed to make use of these services in their applications.
    3. The Android framework includes the following key services
        a. Activity Manager − Controls all aspects of the application lifecycle and activity stack.
        b. Content Providers − Allows applications to publish and share data with other applications.
        c. Resource Manager − Provides access to non-code embedded resources such as strings, color settings and user interface layouts.
        d. Notifications Manager − Allows applications to display alerts and notifications to the user.

e. View System − An extensible set of views used to create application user interfaces.

- **Android Runtime and Core Libraries:**
    1. Android currently uses Android Runtime (ART) to execute application code.
    2. ART is preceded by the Dalvik Runtime that compiled developer code to Dalvik Executable files (Dex files). These execution environments are optimized for the android platform taking into consideration the processor and memory constraints on mobile devices.
    3. Some of the core libraries that are present in the Android operating systems are:
        a. android.app − Provides access to the application model and is the cornerstone of all Android applications.
        b. android.content − Facilitates content access, publishing and messaging between applications and application components.
        c. android.database − Used to access data published by content providers and includes SQLite database management classes.
        d. android.opengl − A Java interface to the OpenGL ES 3D graphics rendering API.
        e. android.os − Provides applications with access to standard operating system services including messages, system services and inter-process communication.
        f. android.text − Used to render and manipulate text on a device display.

g. android.view − The fundamental building blocks of application user interfaces.
h. android.widget − A rich collection of pre-built user interface components such as buttons, labels, list views, layout managers, radio buttons etc.
i. android.webkit − A set of classes intended to allow web-browsing capabilities to be built into applications.

- **Linux Kernel:**

1. The root component of the Android System is the Linux Kernel. It is the foundational piece that enables all of Android's functionality.
2. The Linux Kernel is a battle-tested piece of software that has been used in developing operating systems for devices of wide range, from supercomputers to small gadgets. It has limited processing abilities like small networked gadgets for the Internet of Things (IoT).
3. The Linux Kernel can be tweaked to meet the device specifications to make it possible for manufacturers to make Android devices with different capabilities to match user experience.

Four Kinds of Android Components:

Four Kinds Of Android Components are as follows:

- **Service:**

1. A service in Android is a background process.
2. Services are typically used for processes that are ongoing or that take a significant period of time.

3. A service doesn't have a user interface, so they are often combined with other components such as activities.
4. A typical example is an app in which an activity starts a service running on user interaction, with the service perhaps uploading data to a web resource.
5. The user can continue to interact with the activity while the service runs because it executes in the background.

- **Content Providers:**
    1. A content provider component supplies data from one application to others on request. Such requests are handled by the methods of the ContentResolver class.
    2. The data may be stored in the file system, the database or somewhere else entirely.
    3. A content provider is implemented as a subclass of ContentProvider class and must implement a standard set of APIs that enable other applications to perform transactions.
- **Broadcast Receivers:**
    1. The Android system makes various types of broadcasts an app can respond to.
    2. Apps can be developed to make these broadcasts, but this is far less likely than to listen for existing broadcasts, at least for first apps.
    3. System announcements include information about the device's hardware, such as the battery level, the screen shutting off, the charger being plugged into an outlet, etc.

4. To receive broadcast announcements on Android, apps can use a broadcast receiver.
5. A typical example of this is a battery level widget in which you want to update the display when the battery level changes. In this case, you could use a service class in conjunction with a broadcast receiver to let your app keep listening for announcements in the background.

- **Activities:**
    1. An activity represents a single screen with a user interface,in-short Activity performs actions on the screen.
    2. For example, an email application might have one activity that shows a list of new emails, another activity to compose an email, and another activity for reading emails. If an application has more than one activity, then one of them should be marked as the activity that is presented when the application is launched.

○ Manifest.xml File:

1. Every app project must have an AndroidManifest.xml file (with precisely that name) at the root of the project source set.
2. The manifest file describes essential information about your app to the Android build tools, the Android operating system, and Google Play.
3. On Android Studio to build app, the manifest file is created for already, and most of the essential manifest elements are added to build the app.
4. Among many other things, the manifest file is required to declare the following:
    a. The app's package name, which usually matches your code's namespace. The Android build tools use this to determine the location of code entities when building your project. When

packaging the app, the build tools replace this value with the application ID from the Gradle build files, which is used as the unique app identifier on the system and on Google Play.

b. The components of the app, which include all activities, services, broadcast receivers, and content providers. Each component must define basic properties such as the name of its Kotlin or Java class. It can also declare capabilities such as which device configurations it can handle, and intent filters that describe how the component can be started.
c. The permissions that the app needs in order to access protected parts of the system or other apps. It also declares any permissions that other apps must have if they want to access content from this app.
d. The hardware and software features the app requires, which affects which devices can install the app from Google Play.

B) Android Activity Life cycle:

1. Activities in the system are managed as activity stacks.
2. When a new activity is started, it is usually placed on the top of the current stack and becomes the running activity the previous activity always remains below it in the stack, and will not come to the foreground again until the new activity exits. There can be one or multiple activity stacks visible on screen.
3. An activity has essentially four states:
    a. If an activity is in the foreground of the screen (at the highest position of the topmost stack), it is active or running. This is usually the activity that the user is currently interacting with.
    b. If an activity has lost focus but is still presented to the user, it is visible. It is possible if a new non-full-sized or transparent activity has focus on top of your activity, another activity has higher position in multi-window mode, or the activity itself is not focusable in current windowing mode. Such activity is completely alive (it maintains all state and member information and remains attached to the window manager).
    c. If an activity is completely obscured by another activity, it is stopped or hidden. 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.
    d. The system can drop the activity from memory by either asking it to finish, or simply killing its process, making it destroyed. When it is displayed again to the user, it must be completely restarted and restored to its previous state.
4. The following diagram shows the important state paths of an Activity. The square rectangles represent callback methods you can implement to perform operations when the Activity moves between states. The colored ovals are major states the Activity can be in.

onCreate()

onStart()

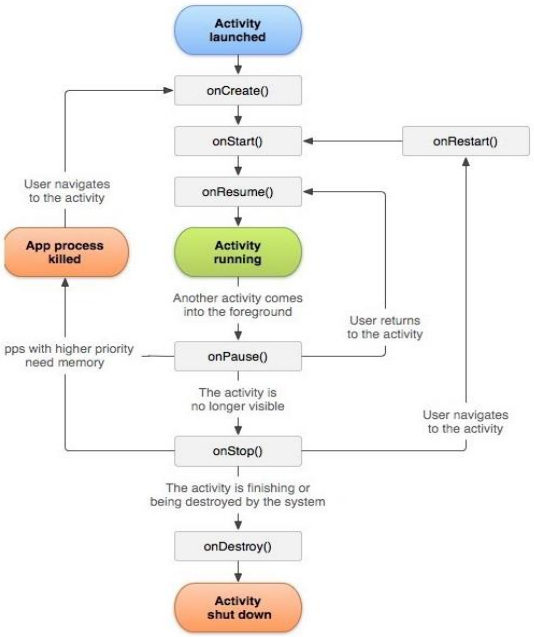
onRestart()

onResume()

onPause()

onStop()

onDestroy()



5. There are three key loops you may be interested in monitoring within your activity:
    a. The entire lifetime of an activity happens between the first call to onCreate(Bundle) through to a single final call to onDestroy(). An activity will do all setup of "global" state in onCreate(), and release all remaining resources in onDestroy(). For example, if it has a thread running in the background to download data from the network, it may create that thread in onCreate() and then stop the thread in onDestroy().
    b. The visible lifetime of an activity happens between a call to onStart() until a corresponding call to onStop(). During this time the user can see the activity on-screen, though it may not be in the foreground and interacting with the user. Between these two methods you can maintain resources that are needed to show the activity to the user. For example, you can register a BroadcastReceiver in onStart() to monitor for changes that impact your UI, and unregister it in onStop() when the user no longer sees what you are displaying. The onStart() and onStop() methods can be

called multiple times, as the activity becomes visible and hidden to the user.

c. The foreground lifetime of an activity happens between a call to onResume() until a corresponding call to onPause(). During this time the activity is in visible, active and interacting with the user. An activity can frequently go between the resumed and paused states -- for example when the device goes to sleep, when an activity result is delivered, when a new intent is delivered -- so the code in these methods should be fairly lightweight.

6. The entire lifecycle of an activity is defined by the following Activity methods. All of these are hooks that you can override to do appropriate work when the activity changes state. All activities will implement onCreate(Bundle) to do their initial setup; many will also implement onPause() to commit changes to data and prepare to pause interacting with the user, and onStop() to handle no longer being visible on screen.