Android is a mobile operating system based on the LINUX Kernel - another open source operating system. It was developed by Android Inc. Android Inc. was founded in California by Andy Rubin, Rich Miner, Nick Sears and Chris White. Later Google acquired the company.

Android Apps are basically software applications that can run on the Android Platform. Apps are mostly written in JAVA Programming Language.

**Android Software Development -**

It is essentially the process by which new Android apps are created. These apps can be written in multiple languages - Kotlin, C++, JAVA employing the Android Software Development Kit (SDK). One can use non JVM languages, but then they need the support of JVM language code.

**Android SDK -**

It basically is a set of tools made for development. It includes tools like - debugger, emulator, documentation for Application Programming Interfaces - APIs etc.

**Android Developer Tools -**

It is a plugin for Eclipse. It is used extensively for developing Android applications. It provides access to many SDK tools as well as a User Interface design tools for designing and building application user interfaces. Eclipse is an IDE - Integrated Development Environment used in computer programming. It is written mostly in JAVA as mentioned that SDK mostly uses JAVA programming to develop apps. But eclipse can be used to develop apps using other languages also but then they demand certain plug-ins to do so.

**Activity Cycle of an Android Application -**

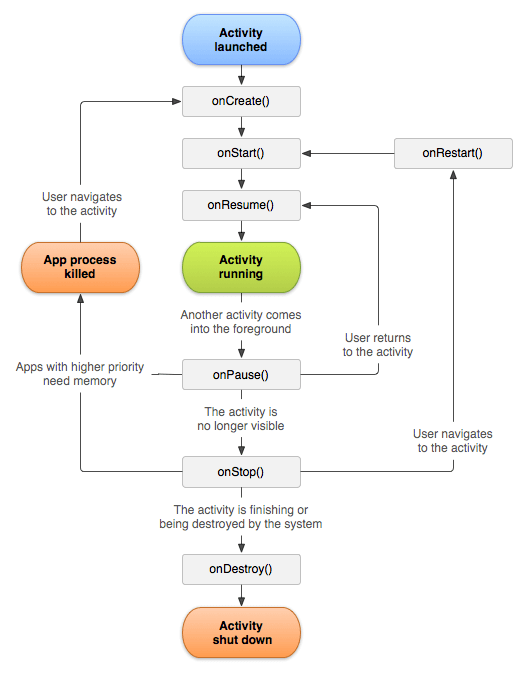
**LIfecycle Callbacks -**

This means to define the activity that needs to be performed when a user exits and re-enters an app. It’s implementation can vary as per need.

The system invocations involve 6 callbacks namely, onCreate(), onStart(), onResume(), onPause(), onStop(), onDestroy(), these are called as per requirements of the transition state activity to be performed.

1. onCreate() - This is done once in the life cycle of the entire application activity. It involves basic startup logic and associated activities.
2. onStart() - When the user starts to use the app, this callback is invoked that makes the app visible to the user in the foreground.
3. onResume() - Once the app appears in the foreground, then this callback is invoked that makes the app actually interactive to the user. The app remains in this state unless the user initiated some other app that forces this to go the background.
4. onPause() - When an interruption disrupts the onResume() state, then this callback is invoked to force the app to enter the Paused state. It may happen that the user wants to go back to the app and use it again, then the onResume() is invoked again along with necessary initialisations released by the onPause() state.
5. onStop() - This is invoked when the user actually is not able to see the activity of the app. In this case, all CPU intensive processes are shut down.
6. onDestroy() - This is called when the activity needs to be destroyed i.e. it needs to be terminated.

There is a minute difference between the onStop() and the onPause() state of the system in the sense that onPause() is called when the some other activity comes into foreground but the current activity is also visible but onStop() is invoked when the current activity is no longer visible to the user.



There are several UI elements/ UI controls involved in any Android app. Following are brief descriptions of five of them -

1. Text View - It is a control that is used to display the text to the user. It doesn’t allow the user to edit the text.
2. EditText - TextView displays the content to the user but doesn’t allow him to edit the text whereas EditText allows the user to enter or modify text. The type of text accepted depends on the type of field used to specify that text.
3. Button - It is control that is responsible for performing actions whenever the user clicks on the button. There can be various types of buttons - ImageButton, ToggleButton, RadioButton etc.
4. Dialogs - It is a small window that prompts messages to the user. It may ask the user to make a decision or to enter additional details. They can be again of several types - AlertDialog, DatePickerDialog, TimePickerDialog
5. Switch - It is basically a representative of a boolean field. It has only two possible states - ON or OFF. It is generally used to toggle some settings On or OFF as per checked or unchecked states.

Insti App is also written in JAVA programming language.

The salient features of JAVA are -

1. It is a fully object oriented programming language.
2. It supports functional programming.
3. It is robust in the sense that it has a much easier to figure out kind of memory management system.
4. It is multi-threaded that means that it has the ability to build apps that require multiple concurrent threads of activities. Therefore it can help in developing highly interactive apps.
5. JAVA is compiled to intermediate form i.e. Bytecode that gives it an ability to be run anywhere once compiled.

Basically, a lot of JVM languages share features with JAVA.

JAVA and C++ have strikingly similar and varying properties too.

**Differences -**

1. Unlike JAVA, C++ is a platform dependent language, because C++ compilers compile the code into machine language whereas JAVA compilers convert it into bytecode.
2. C++ supports pointer mechanism while JAVA doesn't.
3. C++ in some sense is a semi-OOP language whereas JAVA is a fully object-oriented language.
4. Multiple Inheritance is supported in C++ but not in JAVA.

**Similarities -**

1. From the syntax perspective, the two languages are quite similar.
2. They have similar data types, loops, conditionals etc.
3. Execution of the code starts from the main function in both of these languages.

**Android Layouts -**

The user interface is made up of what is called a View object that is an instantiation of the class View. Views are the base classes for all the UI components mentioned above.

The ViewGroup is a subclass of View. Multiple views can be clubbed together to provide android layout which is responsible for the appearance of any app.

**Types -**

**Linear Layout -**

It aligns all children in a single direction either vertically or horizontally. The direction of alignment can be specified in the attribute - android:orientation. Since stacking of elements is one over the other, a vertical list will have one member per row and a horizontal list will have one row of members.

There are various attributes that can be assigned in a linear layout to make the page more informative and attractive.

1. android:id - This is the ID which uniquely determines the layout.
2. android:orientation - It specifies the direction of arrangement i.e. vertical or horizontal. The default value being horizontal.
3. android:layout\_weight - This assigns weights to the views which is a representative of the importance of that view that basically indicates how much space it must occupy on the screen.
4. android:layout\_height, android:layout\_width - These are set to appropriate parameters to make an equal distribution of space between the children.

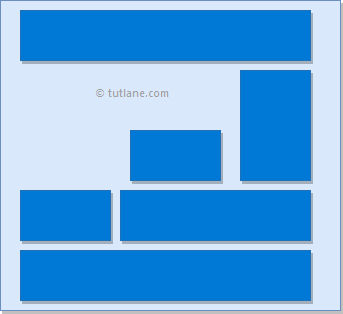
**Relative Layout -**

It displays child views in relative positions wrt each other to the parent.

It is useful in some sense that it can eliminate the usage of nested view groups which helps in improvement of performance of the application.

Again there are many layout properties which can be parameterized according to one’s needs.

1. android:layout\_above - It accepts another sibling view id and places the view above the specified view ID.
2. android:layout\_below - It accepts another sibling view id and places the view below the specified view ID.
3. android:layout\_centerVertical - If it is specified to be True, the view will be vertically centre aligned within its parent.
4. android:layout\_centerInParent - If specified True, the view will be aligned to the centre of the parent.



Relative Layout Linear Layout

**The Dashboard of the App - A Brief Description -**

1. There would be a button with text written on it as Login/Sign Up. Clicking this would direct the user to a page where he can enter his/her details to either sign up or login as per the situation.
2. There would be a profile pic button on which the user can click to see if there are any pending requests for the books he made available on the app and accept them or reject them as per his choice.
3. There would be a Notifications icon on the left that would also be a button that leads to the Notifications section that would give the user notifications if anyone requests borrowing of the user’s books. Also, there will be notifications showing up for the updates subscribed to for the availability of a particular book that he/she searched for recently.
4. Then the main area on the screen would be devoted to the books available on the app. They would initially be sorted on the basis of more recently made available.
5. A left swipe on the screen would lead to a filters section where the user can select the desired filters to suit his/her interests. Then appropriately filtered results would only be shown to the user.
6. Now, for each book the details that would be made available on the main screen would comprise of the book photograph, author, area of the person’s address who has the book.
7. Now, each book displayed on the screen would itself be a button which when clicked will give the complete information about the book so that the user can decide as to whether he wishes to take it or not.
8. There will be Report Problem highlighted text in the bottom of the screen where a user can report any issue to the developer if he/she is facing any issues.

The various layouts identifiable are -

1. The filters section will be linear layout of nested linear layouts because each filter will have a list of options of its own that will be a linear layout.
2. The books displayed will again be a linear layout as the views are stacked one over the other.
3. The search bar, notifications button, the profile menu and the login button would be again in a linear layout as they are arranged in on row.
4. Now, the various sections that are part of the screen will be in a relative layout as their location on the screen is relative to one another and the parent view that is the complete screen.

The UI elements used are -

1. TextView (Books, Filter options)
2. Buttons (Almost everywhere - Radio buttons, Image Buttons)
3. EditText (Search bar)
4. Dialog Boxes (Notifications Section)