

Development of Android Application



एंड्राइड क्या है?

एंड्राइड एक Operating System है जो की linux kernel के ऊपर आधारित है। Android Operating System का इस्तमाल आप अपने SmartPhones में होते हुए देख सकते हैं। अगर मैं इसे आसान भाषा में कहूँ तो Linux एक operating system हैं जिसे की मुख्यतः server और desktop computer में इस्तमाल होता है। तो Android बस एक version है Linux का जिसे की बहुत सारे modification के बाद बनाया गया है।

Android एक ऐसा Operating System है जिसे की design किया गया था Mobile को नज़र में रखते हुए. ताकि इसमें phone की सारी functions और applications को आसानी से run किया जा सके।

आप जो कुछ भी phone के display में देखते हैं वो सारे operating system के ही भाग हैं. जब भी आप कोई call, text message या email पाते हैं तब आपकी OS उसे process करती हैं और आपके सामने readable format में पेश करती है।

Android एक बेहतरीन Mobile Operating System

Android एक ऐसा बेहतरीन Mobile Operating System है जिसे की Google द्वारा बनाया गया है, देखा जाये तो Google द्वारा बनायीं गयी Software को आज दुनिया में प्राय सभी Mobile Phones में इस्तमाल किया जाता है. केवल Apple's iPhones को छोड़कर।

Android एक Linux-based software system है. जैसे की Linux एक Open Source software है और इसके साथ ये बिलकुल Free भी है. इसका मतलब ये है की दुसरे Mobile Company भी Android Operating Systems का इस्तमाल कर सकते हैं. इसमें जो distinguishing factor हैं वो हैं की इस Brand की kernel।

Android के Central Core को host करता है जो की essentially एक strip code है और जो की Software को operate होने में मदद करता है।

Android के अलग अलग Versions

- Android 1.0 Alpha
- Android 1.1 Beta
- Android 1.5 Cupcake
- Android 1.6 Donut
- Android 2.1 Eclair
- Android 2.3 Froyo
- Android 2.3 Gingerbread
- Android 3.2 Honeycomb
- Android 4.0 Ice Cream Sandwich
- Android 4.1 Jelly Bean
- Android 4.2 Jelly Bean
- Android 4.3 Jelly Bean
- Android 4.4 KitKat
- Android 5.0 Lollipop
- Android 5.1 Lollipop
- Android 6.0 Marshmallow
- Android 7.0 Nougat
- Android 7.1 Nougat
- Android 8.0 Oreo
- Android 8.1 Oreo
- Android 9.0 Pie
- Android 10
- Android 11
- Android 12
- Android 13
- Android 14

Android की विशेषताएं

1. Open Source

Android Linux kernel पर आधारित है और यह Android Open Source Project के रूप में जाना जाता है। यह Developers और हार्डवेयर निर्माताओं को ऑपरेटिंग सिस्टम के Core Software में बदलाव करने की अनुमति देता है। जिस कारण सभी मोबाइल निर्माता अपने हिसाब से सॉफ्टवेयर को Customize कर सकते हैं।

2. Connectivity

Android GSM/EDGE, IDEN, CDMA, EVDO, UMTS, Bluetooth, Wi-Fi, LTE, NFC और WiMAX के साथ-साथ और भी Connectivity Technology को सपोर्ट करता है।

3. Multi language

Android अलग-अलग भाषाओं को भी सपोर्ट करता है जैसे अंग्रेजी, जर्मन, चीनी, जापानी, कोरियाई, रूसी, हिंदी इंडोनेशियाई, स्पेनिश, तुर्की इत्यादि। आप अपनी सुविधा के अनुसार कोई भी भाषा चुन सकते हैं।

4. Multitasking

Multi Tasking का मतलब होता है कि आप एक साथ कई अलग-अलग चीज़ें कर सकते हैं। Android की यह सुविधा बहुत उपयोगी है। इसकी मदद से यूजर एक साथ कई एप्लीकेशन चला सकते हैं।

5. User interface

Android ऑपरेटिंग सिस्टम का यूजर इंटरफ़ेस हमारे काम को बहुत आसान बनाता है क्योंकि Android स्मार्टफोन में आपको Apps खोजने की ज़रूरत नहीं होती है। यह आपके Main Screen पर ही दिखाई देता है।

6. Security updates

गूगल समय-समय पर OTA Updates के माध्यम से यूजर्स को Bug Fixe और Android Security Update प्रदान करता रहता है।

7. Multi Touch

multi-Touch का मतलब है कि आप अपने स्मार्टफोन को दोनों उंगलियों से चला सकते हैं जैसे आप अपने किसी फोटो पर Pinch Zoom कर सकते हैं। इस सुविधा को सबसे पहले HTC Hero जैसे Handset में उपलब्ध कराया गया था।

8. Media support

Android विभिन्न प्रकार के ऑडियो और वीडियो का सपोर्ट करता है जैसे H.263, H.264, MPEG-4 SP, AMR, AMR-WB, AAC, HE AAC, AAC 5.1, MP3, MIDI, Ogg Vorbis, WAV, JPEG, PNG, GIF, BMP इत्यादि।

9. Screen Capture

Android पर एक ही समय में पावर और होम स्क्रीन बटन दबाकर Screenshot Capture करने के लिए सपोर्ट करता है।

Android के लाभ

- Android को कोई भी डेवलप कर सकता है।
- Android Apps तक पहुंचना बहुत आसान है।
- यह सभी गूगल सेवाओं को सपोर्ट करता है।
- Android आपको नए SMS और Email या नए अपडेट के बारे में सूचित करता है।
- Android फोन इंटरनेट Share करने के लिए Router के रूप में भी काम कर सकता है।
- यह 2D एंड 3D Graphic को सपोर्ट करता है।
- इसमें हम लाखों Apps Install कर सकते हैं।
- Android ऑपरेटिंग सिस्टम में Apps का बैकअप है।
- यह Third Party Apps को भी सपोर्ट करता है।
- इसमें अलग-अलग ऐप एक ही समय में चल सकते हैं।

Android के नुकसान

- Android OS में App Background में चालू रहते हैं। जिससे बैटरी और डाटा की खपत ज्यादा तेजी से होती है।
- इसमें गूगल अकाउंट की आवश्यकता होती है।
- इसकी Apps के भीतर कई सारे विज्ञापन होते हैं।
- कुछ Apps की Quality अच्छी नहीं होती है।
- Android फोन में कई ऐप पहले से Install होते हैं जिन्हें आप Uninstall नहीं कर सकते हैं।

Android AOSP Ka Architecture

एंड्रॉइड एओएसपी आर्किटेक्चर एक layered architecture है जिसमें कई कॉम्पोनेन्ट शामिल हैं

AOSP Architecture के 5 मुख्य कंपोनेंट्स हैं:-

- Linux Kernel
- Libraries
- Android Runtime
- Application Framework
- System Applications

Applications

Home

Contacts

Phone

Browser

Camera

Application Framework

Activity
Manager

Window
Manager

Content
Providers

View
System

Package
Manager

Telephony
Manager

Resource
Manager

Location
Manager

Notification
Manager

Libraries

Surface
Manager

Media
Framework

SQLite

OpenGL | ES

FreeType

Webkit

SGL

SSL

Libe

Android Runtime

Libraries Core

Dalvik Virtual
Machine

Linux Kernel

Display
Driver

Camera
Driver

Flash Memory
Driver

Binder (IPC)
Driver

Keypad
Driver

WiFi
Driver

Audio
Driver

Power
Management

1. Linux Kernel

लिनक्स कर्नल एंड्रॉइड आर्किटेक्चर की सबसे निचली, महत्वपूर्ण और मुख्य लेयर है। यह सुरक्षा, मेमोरी मैनेजमेंट, मल्टीटास्किंग, डिवाइस मैनेजमेंट और प्रक्रिया मैनेजमेंट जैसी वेरियस फीचर्स प्रदान करता है। यह डिवाइस हार्डवेयर और एंड्रॉइड आर्किटेक्चर की ऊपरी लेयर्स के बीच abstract level को बनाए रखने के लिए भी जिम्मेदार है। इसमें कैमरा, कीपैड, ऑडियो, वाई-फाई, डिस्प्ले आदि जैसे ड्राइवर शामिल हैं।

2. Libraries

Android AOSP में यह लेयर एप्लिकेशन चलाने के लिए आवश्यक बेसिक फ्रेम प्रदान करती है। इसमें लाइब्रेरी और एंड्रॉइड रनटाइम का एक सेट शामिल है। एंड्रॉइड component C/C++ में लिखे गए मूल कोड और native लाइब्रेरीज का उपयोग करके बनाया गया है, और अधिकांश लाइब्रेरीज ओपन सोर्स हैं।

यह लेयर हार्डवेयर के लिए विशिष्ट डेटा का मैनेजमेंट भी करती है। कुछ लाइब्रेरी SSL, SQLite, OpenGL आदि हैं।

जैसे जावा JVM का उपयोग करता है, वैसे ही एंड्रॉइड DVM (Dalvik वर्चुअल मशीन) का उपयोग करता है, जो एंड्रॉइड एप्लिकेशन चलाने और बैटरी लाइफ, मेमोरी और प्रदर्शन को अनुकूलित करने के लिए जिम्मेदार है।

3. Android Runtime

एंड्राइड runtime में कोर लाइब्रेरी और DVM हैं जो एंड्राइड एप्लीकेशन को चलाने के लिए जिम्मेदार हैं DVM , JVM की तरह है लेकिन ये मोबाइल उपकरणों के लिए अनुकूलित हैं यह less memory consume करता है और high performance प्रदान करता है

4. Application Framework

एप्लिकेशन फ्रेमवर्क API प्रदान करता है जो डेवलपर्स को एंड्रॉइड ऐप बनाने की अनुमति देता है, साथ ही स्टैण्डर्ड APP कंपोनेंट्स (जैसे एक्टिविटीज, Services और कंटेंट प्रोवाइडर) का एक सेट प्रदान करता है जिसका उपयोग डेवलपर्स अपने APP बनाने के लिए कर सकते हैं। नेटिव लाइब्रेरीज के टॉप पर बनाया गया एप्लिकेशन frame हमें एक एप्लिकेशन प्रोग्रामिंग इंटरफ़ेस प्रदान करता है। एंड्रॉइड ऑपरेटिंग सिस्टम की सभी सुविधाएं हमें API के माध्यम से उपलब्ध होती हैं जो जावा क्लास के रूप में लिखी जाती हैं।

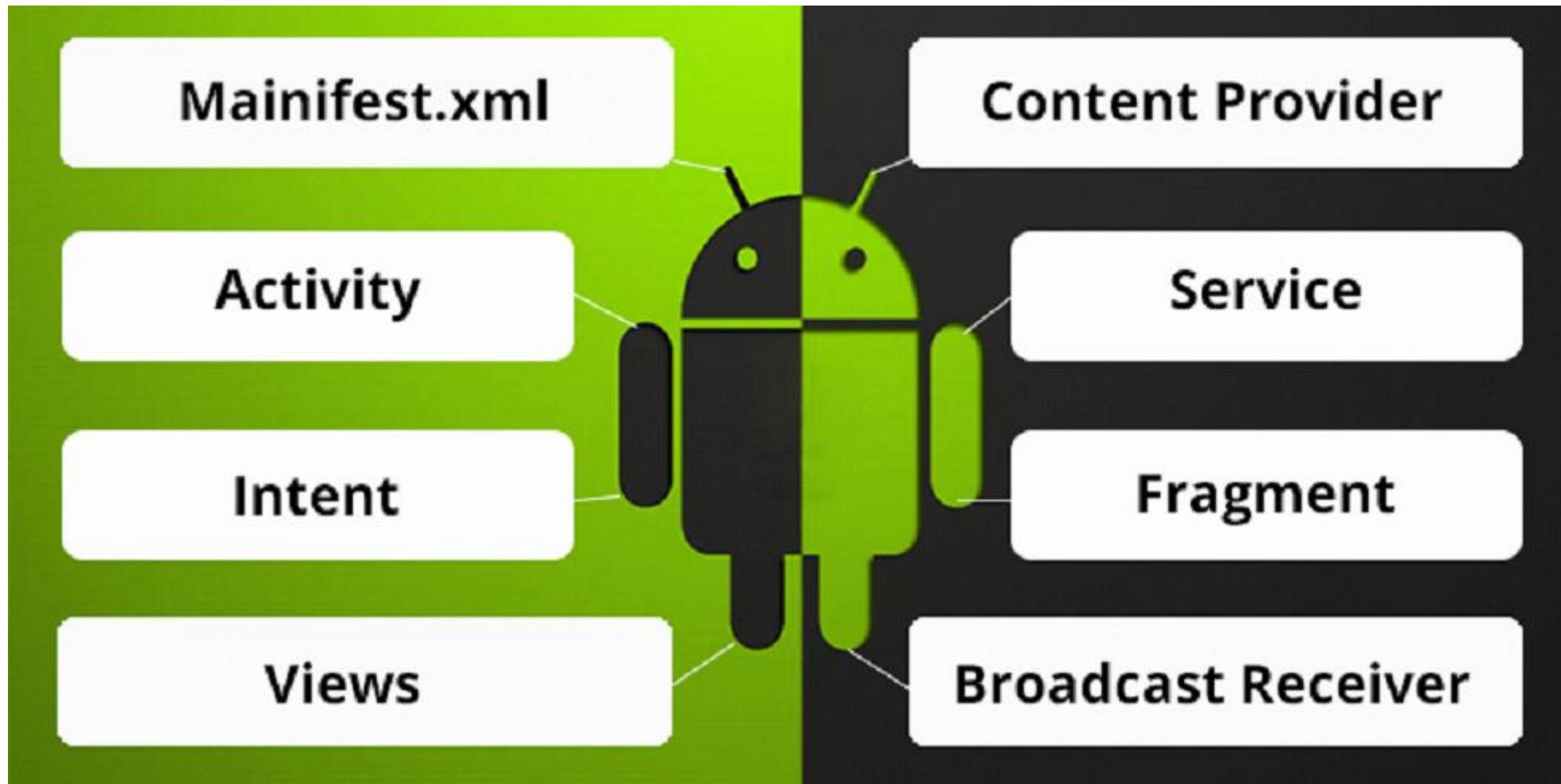
एप्लिकेशन फ्रेमवर्क में हार्डवेयर एब्स्ट्रैक्शन लेयर भी शामिल है, जिसे HAL भी कहा जाता है, जो एप्लिकेशन फ्रेमवर्क को सभी hardware-specific drivers के साथ संचार करने की अनुमति देता है। एक एंड्रॉइड एप्लिकेशन सभी विभिन्न हार्डवेयर टूल्स से कमांड प्राप्त करने के लिए HAL एपीआई का उपयोग करता है।

5. Applications

यह एंड्रॉइड AOSP आर्किटेकचर की सबसे ऊपरी लेयर्स है जो यूजर इंटरफ़ेस है। इस लेयर्स में मूल एंड्रॉइड एप्लिकेशन और third party इंस्टॉल किए गए एप्लिकेशन हैं, और यह उपयोगकर्ता इनपुट को प्रबंधित करने के लिए भी काम करता है।

उन्हें एक पैकेज में एक साथ जोड़ दिया जाता है, और इंस्टॉल किए जाने वाले सभी एप्लिकेशन इस लेयर्स में लिखे जाते हैं, जैसे संपर्क, ब्राउज़र, सेटिंग्स, संदेश, गेम इत्यादि।

ANDROID BASIC BUILDING BLOCKS



- 1.Activities
- 2.Servives
- 3.Broadcast receivers
- 4.Content Provider
- 5.Views
- 6.Intents
- 7.Fragments
- 8.AndroidMainfest.xml

ACTIVITIES

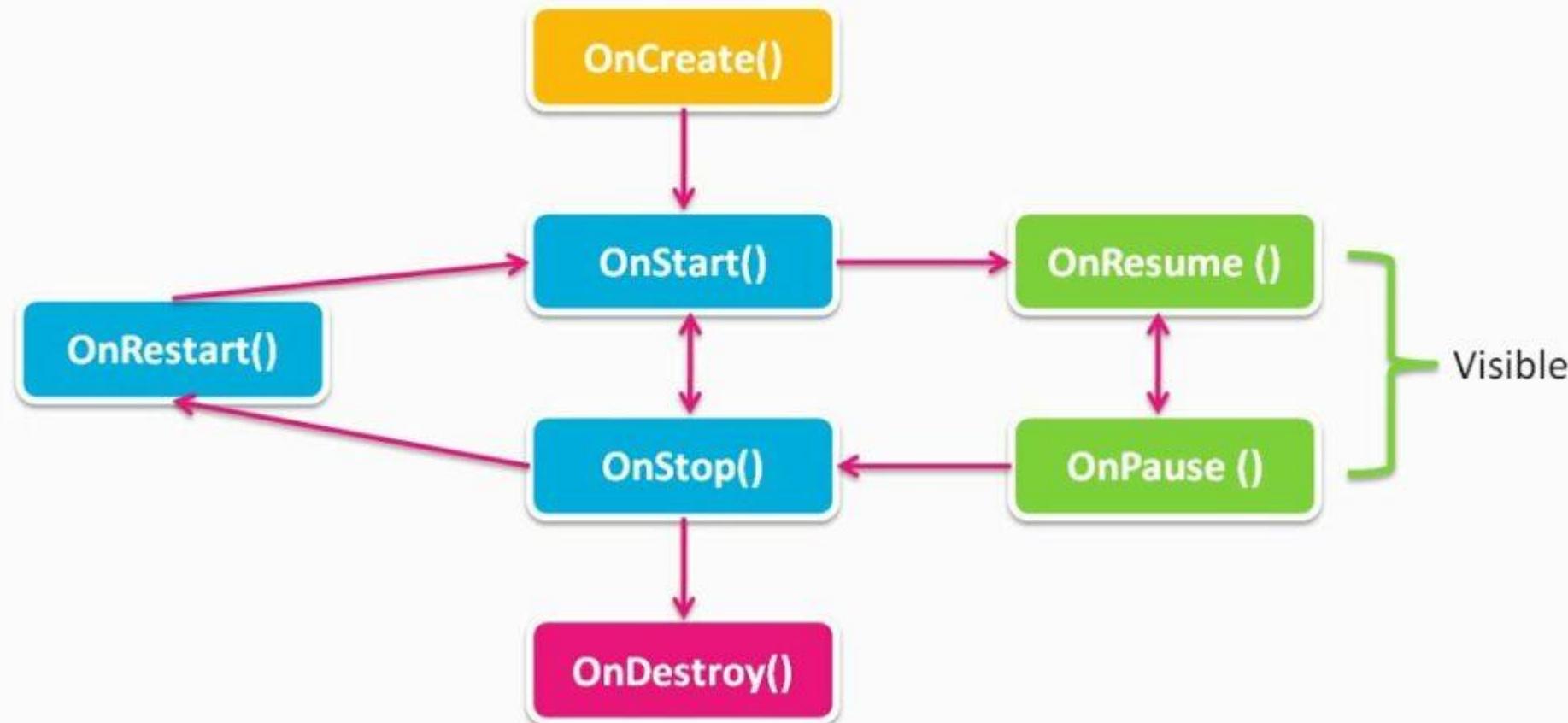
आप की एप्लीकेशन में एक से ज्यादा activity हो सकती है। जब आप इन activities को navigate करते हैं, यानि एक activity से दूसरी activity में जाते हैं तो activity अपनी **life cycle** की different stages में हो जाती है.

example के लिए यदि आप एक video player बना रहे हैं तो जब user किसी दूसरी activity पर जाने पर तो video streaming रुक जानी चाहिए। लेकिन यदि आप एक music player बना रहे हैं तो आप music को background में भी play कर सकते हैं.

```
public class MainActivity extends Activity
```

```
{  
}
```

Activity Lifecycle Methods



- **Created** – इस stage में onCreate() मेथड द्वारा activity create की जा सकती है जब user आपकी application के icon पर क्लिक करता है।
- **Started** – इस stage में onStart() मेथड द्वारा activity start की जाती है।
- **Resumed** – इस stage में activity running stage में होती है और user को visible होती है
- **Paused** – इस stage में activity paused होती है और जब user किसी दूसरी application में switch करता है।
- **Stopped** – इस stage में activity stopped होती है।
- **Destroyed** – इस stage में activity destroyed हो जाती है।

SERVICES

- Service एक facility होती है जिससे आप android system को बता सकते हैं कि आप कोई काम background में करते रहना चाहते हैं।
- Service एक facility होती है जिससे एक एप्लीकेशन अपनी कुछ functionality दूसरी applications को expose कर सकती है।

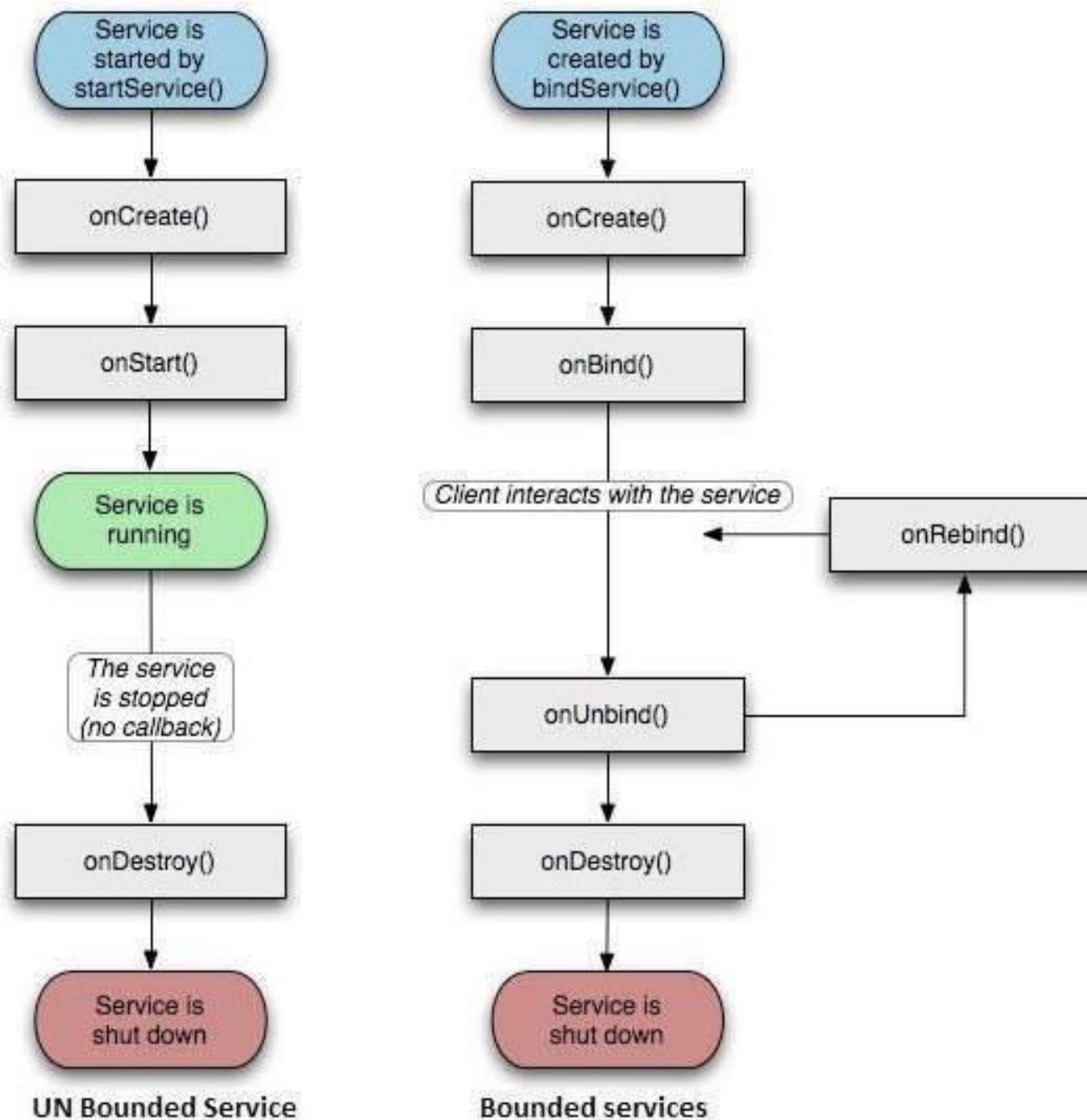
Services दो प्रकार की होती हैं -

- १- LOCAL services – access from within the application
- २- Remote services – accessed remotely from other applications running on the same device.

एक service की 2 State हो सकती है-

- **Started** - एक service started service कहलाती है जब कोई component startService() मेथड कॉल करता है। एक बार स्टार्ट होने के बाद service बैकग्राउंड में execute होती रहेगी चाहे इसे क्रिएट करने वाला component destroy हो जाये। जब operation कम्पलीट हो जाता है तो service automatically stop हो जाती है।
- **Bounded** - एक एप्लीकेशन bound होती है जब कोई component bindService() मेथड कॉल करके service को bind करता है। एक bound service client server interface प्रोवाइड करती है जिससे component service के साथ interact कर सकता है।

Life Cycle of services



•**onStartCommand()** - Android system इस मेथड को तब कॉल करता है जब कोई दूसरा component जैसे की कोई activity startService() कॉल करके service को start करने के लिए request करता है। जैसे ही ये मेथड कॉल होता है service स्टार्ट हो जाती है और बैकग्राउंड में execute होती रहती है। अगर आप इस मेथड को implement करते हैं तो stopService() मेथड कॉल करके service को स्टॉप करना आपकी जिम्मेदारी होती है।

•**onBind()** - Android system ये मेथड तब कॉल करता है जब कोई component bindService() मेथड कॉल करके service के साथ bind होना चाहता है। जब आप इस मेथड को implement करे तो आपको एक interface प्रोवाइड करना चाहिए जिससे clients service से communicate कर सके।

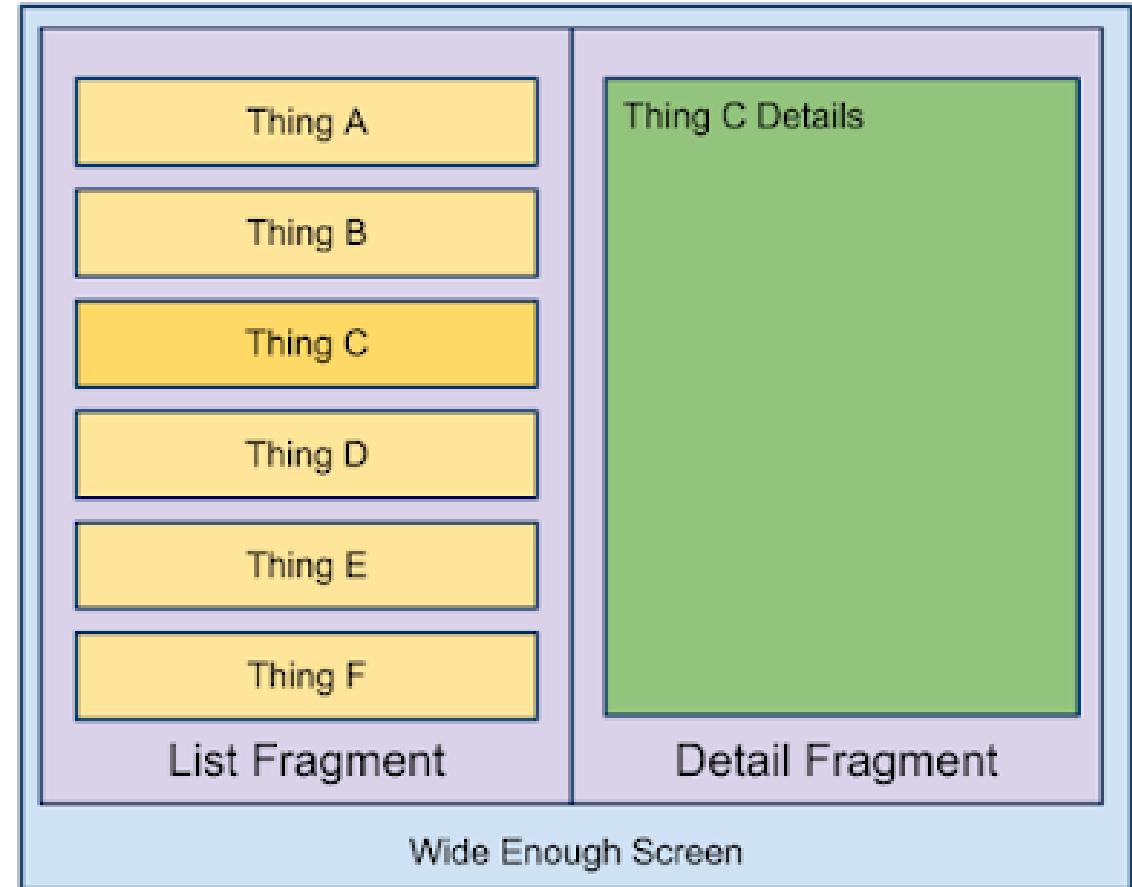
- **onUnbind ()** - सिस्टम इस **method** को तब कॉल करता है जब सभी क्लाइंट सेवा द्वारा प्रकाशित किसी विशेष इंटरफ़ेस से डिस्कनेक्ट हो जाते हैं।
- **onRebind ()** - जब नए **client service** से जुड़ जाते हैं , तो **system** इस **method** को कॉल करता है
- **onCreate()** - **Android system** इस **method** को तब कॉल करता है जब **service** फर्स्ट टाइम क्रिएट होती है। ये मेथड सिर्फ एक बार ही कॉल होता है।
- **onDestroy()** - **Android system** इस मेथड को तब कॉल करता है जब **service** अपना काम पूरा कर लेती है और **destroy** होती है।

Views

Android views एक्टिविटी के अंदर होते हैं। Android views वो elements होते हैं जो हमें स्क्रीन पर शो होते हैं, जैसे कि button, text-box आदि। सभी views activity के अंदर होते हैं।

FRAGMENT

एक fragment किसी activity के अंदर एक separate UI (user interface) होता है। Fragments की खुद की life cycle और input events होते हैं। आप एक fragment को कई activities में यूज़ कर सकते हैं। जब आपकी activity running state में हो तब भी आप fragments को add और remove कर सकते हैं। Fragments को आप sub activity भी कह सकते हैं।



Android fragments को android के Honeycomb version में introduce किया गया था। Android fragments से पहले आप user को एक बार में एक activity ही शो कर सकते थे इसलिए आप स्क्रीन को अलग अलग parts में divide नहीं कर सकते थे।

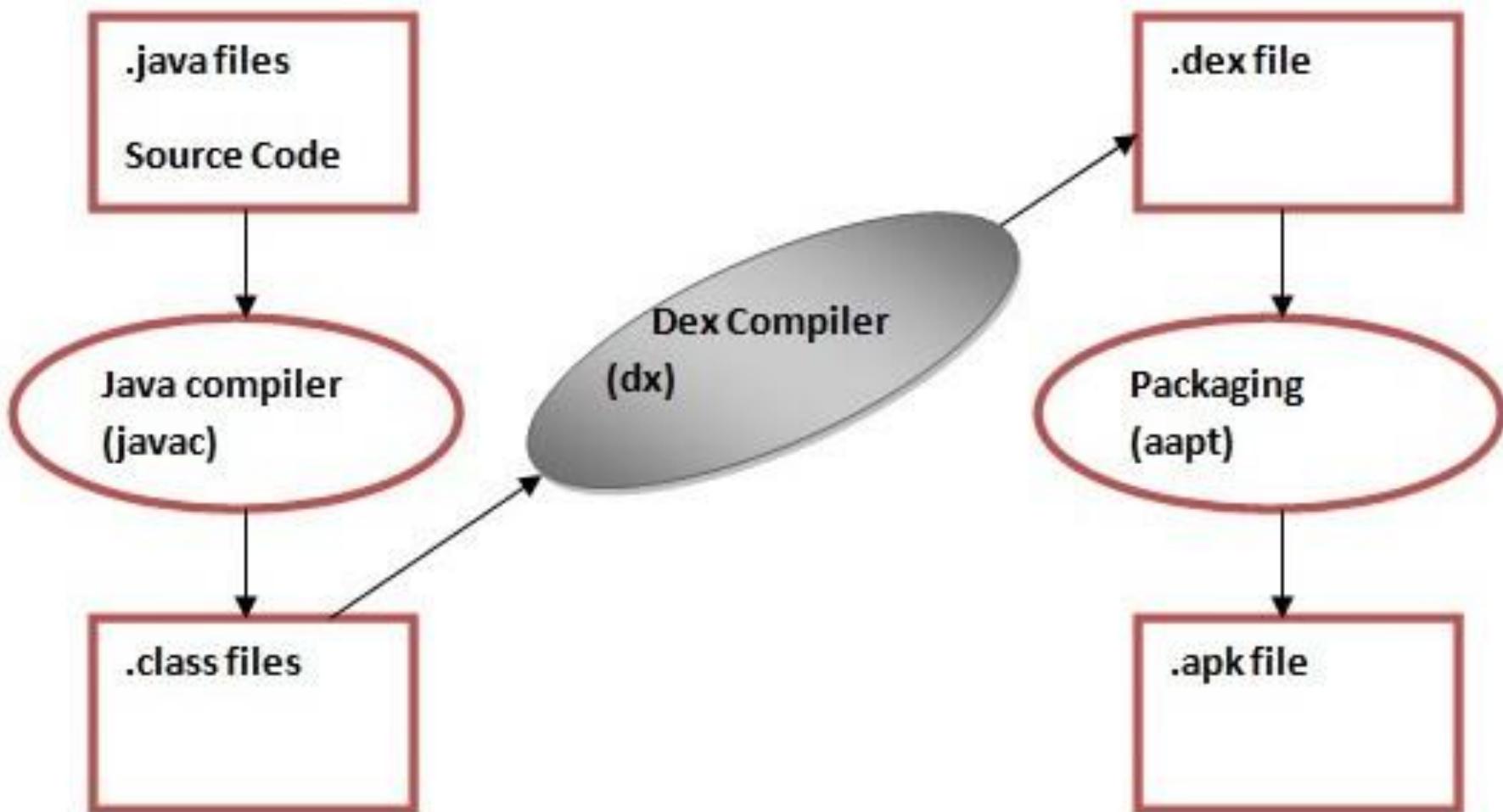
Fragments के आने से application design और भी ज्यादा flexible हो गयी क्योंकि आप यूजर को एक साथ एक से ज्यादा interface शो कर सकते हैं और उन्हें separately manage भी कर सकते हैं। अब आपके पास activity तो एक ही रहेगी लेकिन आप इसमें बहुत से fragments add कर सकते हैं।

Dalvik virtual machine

डाल्विक आइसलैंड के एक कस्बे का नाम है। डेल्विक वीएम डैन बोर्नस्टीन द्वारा लिखा गया था। डेक्स कंपाइलर क्लास फ़ाइलों को .dex फ़ाइल में परिवर्तित करता है जो डाल्विक वीएम पर चलती है। एकाधिक क्लास फ़ाइलें एक डेक्स फ़ाइल में परिवर्तित हो जाती हैं।

Dalvik virtual machine

Dalvik एक ओपन सोर्स, रजिस्टर-आधारित वर्चुअल मशीन(Register-based virtual machine) (VM) है जो Android OS का हिस्सा है। Dalvik VM Dalvik Executable (.dex) प्रारूप में फ़ाइलों को निष्पादित(Execute) करता है और थ्रेडिंग और निम्न-स्तरीय मेमोरी प्रबंधन[Low-level memory management] जैसी अतिरिक्त कार्यक्षमता के लिए लिनक्स कर्नेल पर निर्भर करता है।



APK File क्या है ?

APK का full form होता है - **Android package kit.** APK File गूगल के android मोबाइल ऑपरेटिंग सिस्टम के लिए बनाई गई एक ऐप है। कुछ apps एंड्रॉइड डिवाइस पर पहले से install होकर आते हैं, जबकि अन्य ऐप Google Play Store से download किए जा सकते हैं। गूगल प्ले स्टोर से डाउनलोड किए गए apps आपके एंड्रॉइड डिवाइस पर ऑटोमैटिक रूप से install हो जाते हैं, जबकि अन्य सोर्स से डाउनलोड किए गए APK app को मैन्युअल रूप से इंस्टॉल करना पड़ता है। एपीके को आसान भाषा समझें तो, जिस तरह विंडोज सिस्टम में Softwares को Install करने के लिए .exe फाइल का इस्तेमाल किया जाता है, ठीक उसी तरह एंड्रॉइड के लिए APK File का इस्तेमाल किया जाता है।

इस एंड्राइड पैकेज में single Android program के लिए सभी जरुरी फाइलें स्टोर होती हैं। नीचे सबसे प्रमुख फाइलों और फ़ोल्डरों की सूची देख सकते हैं

•**•META-INF/:** इसमें manifest file, signature एवं resources की लिस्ट होती है।

•**•lib/:** ये एक specific डिवाइस architectures पर run करने वाले Native libraries हैं। जैसे - armeabi-v7a, x86, etc.

•**•res/:** इस Resources फोल्डर में resources.arsc में शामिल नहीं किये रिसोर्स शामिल होते हैं। जैसे - इमेज

- **assets/**: इसमें Raw resource files स्टोर होती है।

- **AndroidManifest.xml**: ये फोल्डर Apk file का नाम, वर्शन आदि बताता है।

- **classes.dex**: डिवाइस में Run करने के लिए Java classes को compiled करने का काम करता है।

- **resources.arsc**: एप्प द्वारा उपयोग किया जाने वाला resources जैसे - strings को compiled करता है।

ANDROID DEVELOPMENT ENVIRONMENT

Android एप्लीकेशन बनाना शुरू करने से पहले एक उपयुक्त डेवलपमेंट एनवायरनमेंट सेट करना आवश्यक है। यह डेवलपर्स के लिए किसी भी एप्लीकेशन को बनाने के लिए आवश्यक उपकरणों का उपयोग करना आसान बनाता है और यह सुनिश्चित करता है कि सभी ऑपरेशन या प्रक्रियाएँ सुचारू रूप से चलें।

एंड्रॉइड application विकसित करने के लिए आवश्यक tools

- i) Java development kit (JDK)
- ii) Android SDK

जावा डेवलपमेंट किट

JDK का पूरा नाम है जावा डेवलपमेंट किट है।

यह एक क्रॉस-प्लेटफॉर्म सॉफ्टवेयर डेवलपमेंट एनवायरनमेंट है जिसमें सॉफ्टवेयर डेवलपमेंट टूल्स और सहायक लाइब्रेरीज का Store होता है। इसका उपयोग जावा आधारित सॉफ्टवेयर Application विकसित करने के लिए किया जाता है और यह जावा का मूल पैकेज है जिसमें Java Runtime Environment (JRE) और Java Virtual Machine (JVM) शामिल हैं।

प्रोग्रामर्स अक्सर सोचते हैं कि क्या JRE केवल जावा प्रोग्राम चलाने के लिए काफी है या क्या उन्हें JDK की आवश्यकता है। उत्तर है कि वास्तविक समय पर चलने वाले Java Applications के लिए जिन्हें जेडके ही प्रदान करता है, उन्हें जटिल सॉफ्टवेयर टूल्किट और लाइब्रेरीज की आवश्यकता होती है।

Android SDK

Android SDK का पूरा नाम Android Software Development Kit होता है।

Android SDK में वो सभी tools होते हैं जिनकी मदद से हम एंड्राइड एप्लीकेशन को create और compile कर सकते हैं।

Android SDK में हमें सोर्स कोड के साथ sample project, development tools , एक इमुलेटर और application को develop करने के लिए लाइब्रेरी मिलती है। सभी एप्लीकेशन java में डेवेलप की जाती है और वर्चुअल मशीन पर रन की जाती है।

To download JDK

<https://www.oracle.com/in/java/technologies/downloads/#jdk22-windows>

To download Android studio

<https://developer.android.com/studio>

चरण 1: एंड्रॉइड स्टूडियो executableया ज़िप फाइल प्राप्त करने के लिए <https://developer.android.com/studio/#downloads> पर जाएं।

चरण 2: डाउनलोड एंड्रॉइड स्टूडियो बटन पर क्लिक करें।

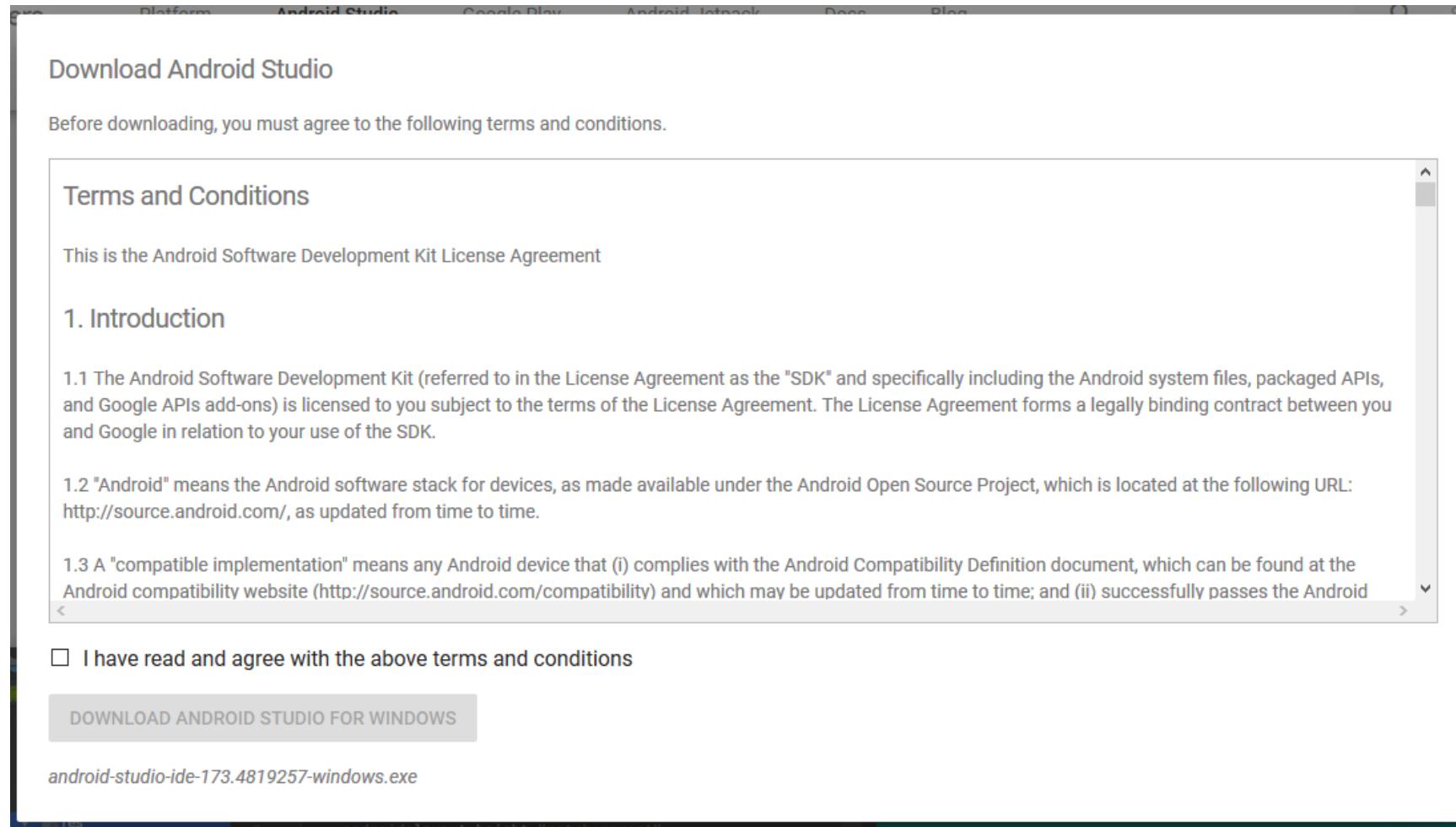


Android Studio provides the fastest tools for building apps on every type of Android device.

[DOWNLOAD ANDROID STUDIO](#)

4.1.3 for Windows 64-bit (896 MiB)

“I have read and agree with the above terms and conditions” चेकबॉक्स पर क्लिक करें,
उसके बाद डाउनलोड बटन पर क्लिक करें।

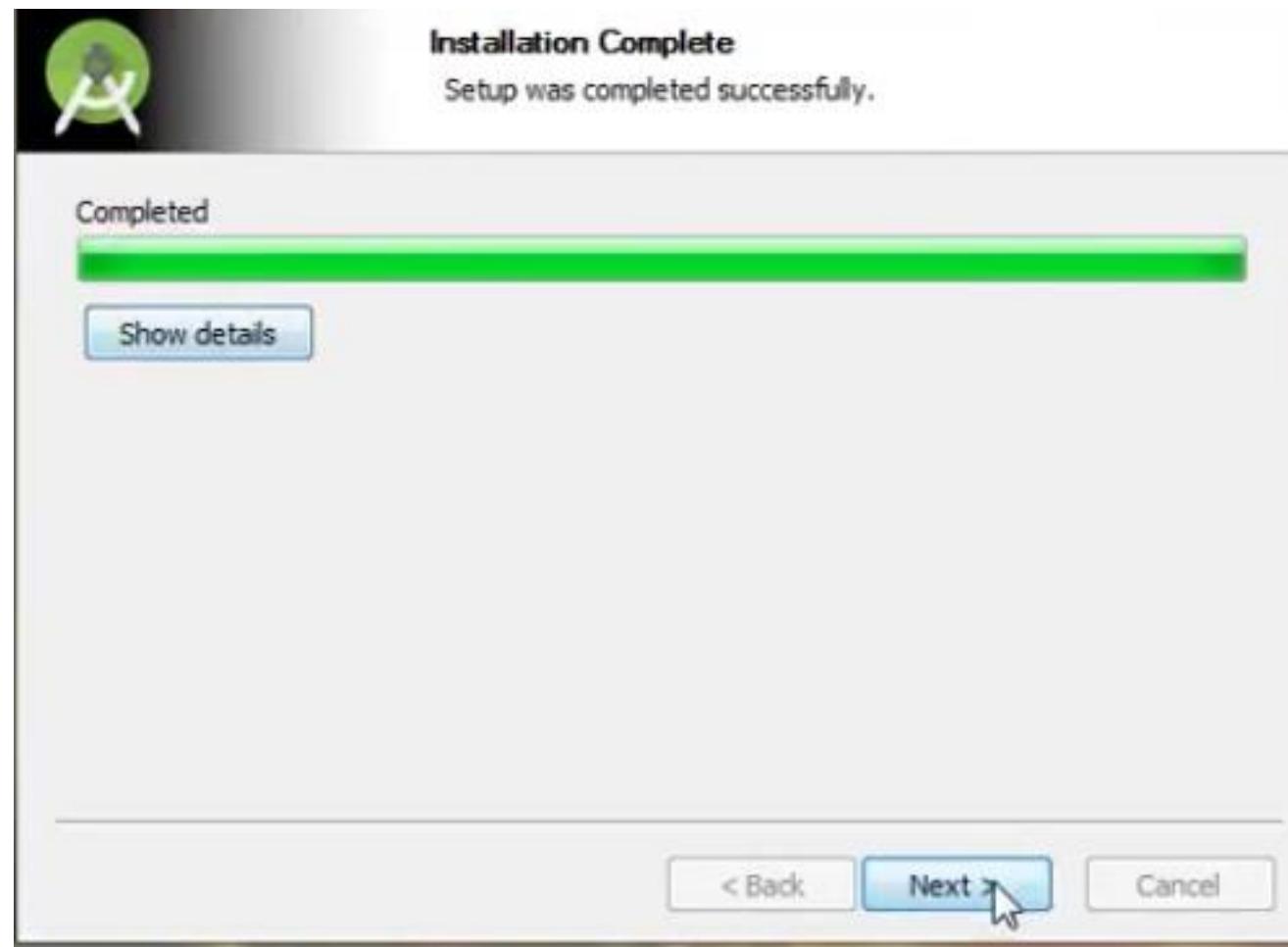


चरण 3: डाउनलोडिंग समाप्त होने के बाद, डाउनलोड से फ़ाइल खोलें और इसे चलाएँ। यह निम्नलिखित डायलॉग बॉक्स को संकेत देगा।

NEXT पर क्लिक करें। अगले प्रॉम्प्ट में, यह इंस्टॉलेशन के लिए पथ पूछेगा। पथ चुनें और अगला पर क्लिक करें।



चरण 4: यह इंस्टॉलेशन शुरू कर देगा, और एक बार यह पूरा हो जाने पर, यह नीचे दिखाए गए चित्र की तरह होगा।





Completing Android Studio Setup

Android Studio has been installed on your computer.

Click Finish to close Setup.

Start Android Studio



< Back

Finish

Cancel

चरण 5: एक बार " समाप्त " पर क्लिक करने के बाद, यह पूछेगा कि क्या पिछली सेटिंग्स को आयात करने की आवश्यकता है [यदि एंड्रॉइड स्टूडियो पहले से इंस्टॉल किया गया था], या नहीं। 'सेटिंग्स आयात न करें' विकल्प चुनना बेहतर है।



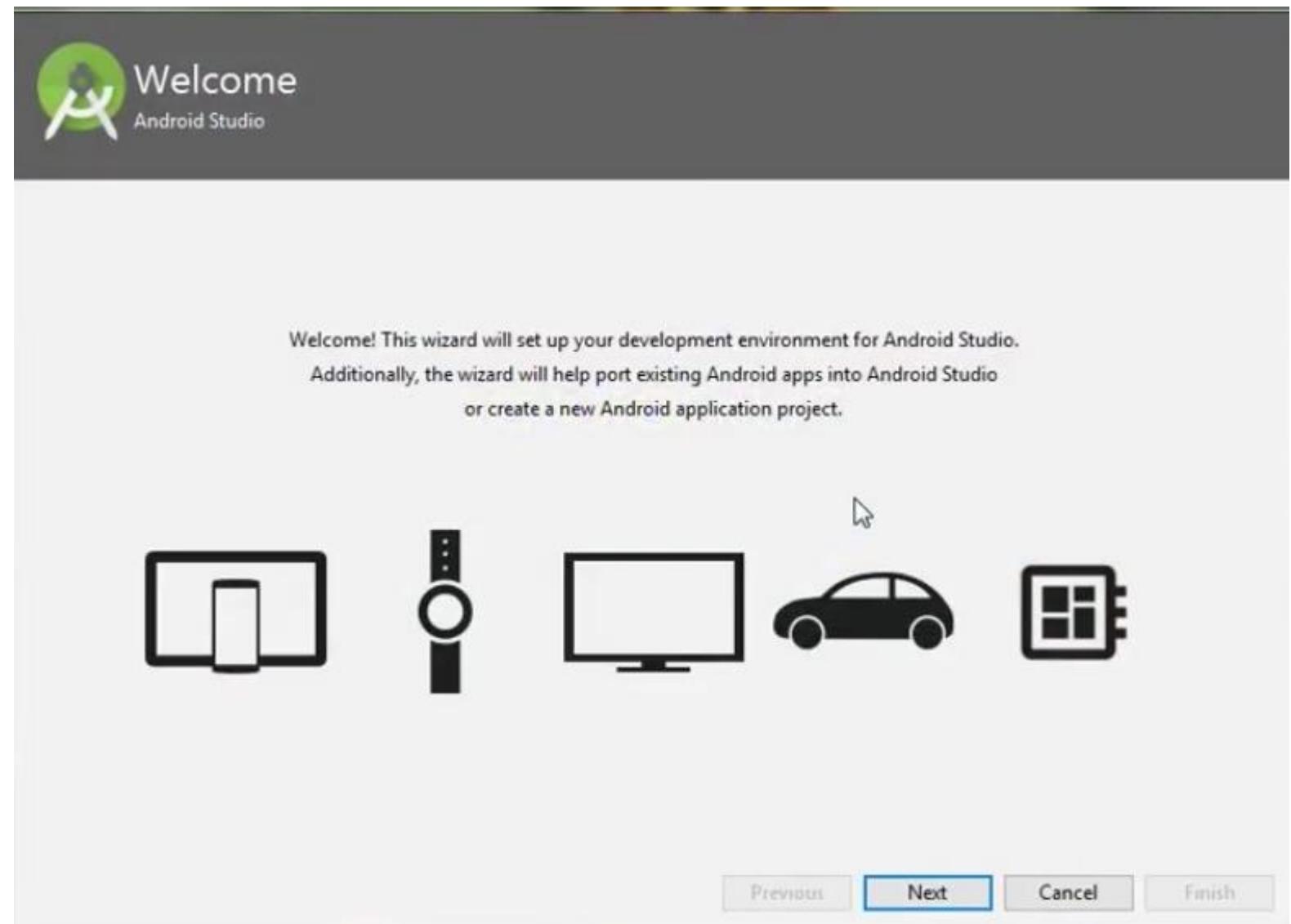
चरण 6: इससे एंड्रॉयड स्टूडियो शुरू हो जाएगा।



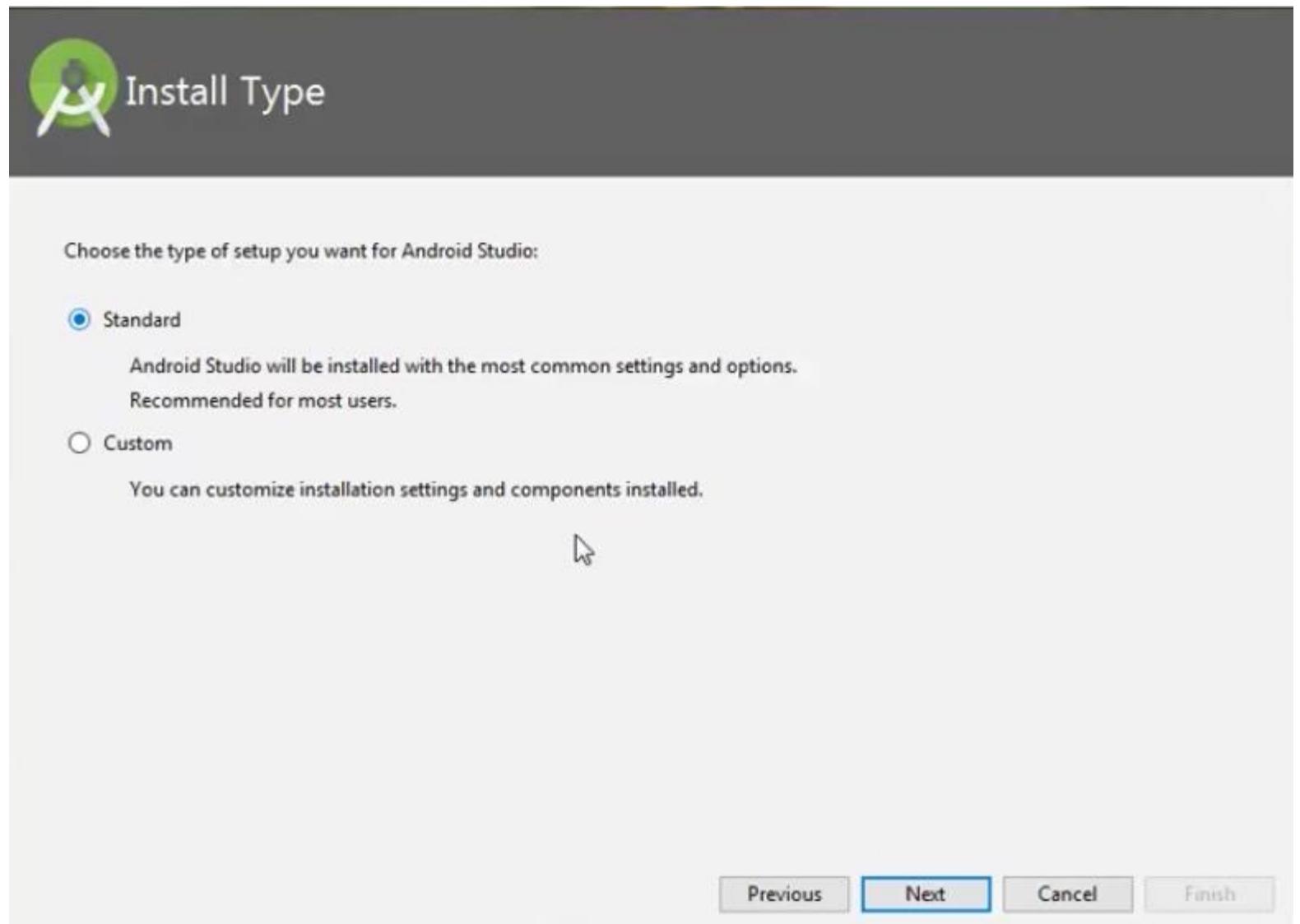
इस बीच, यह उपलब्ध SDK components को ढूँढेगा।



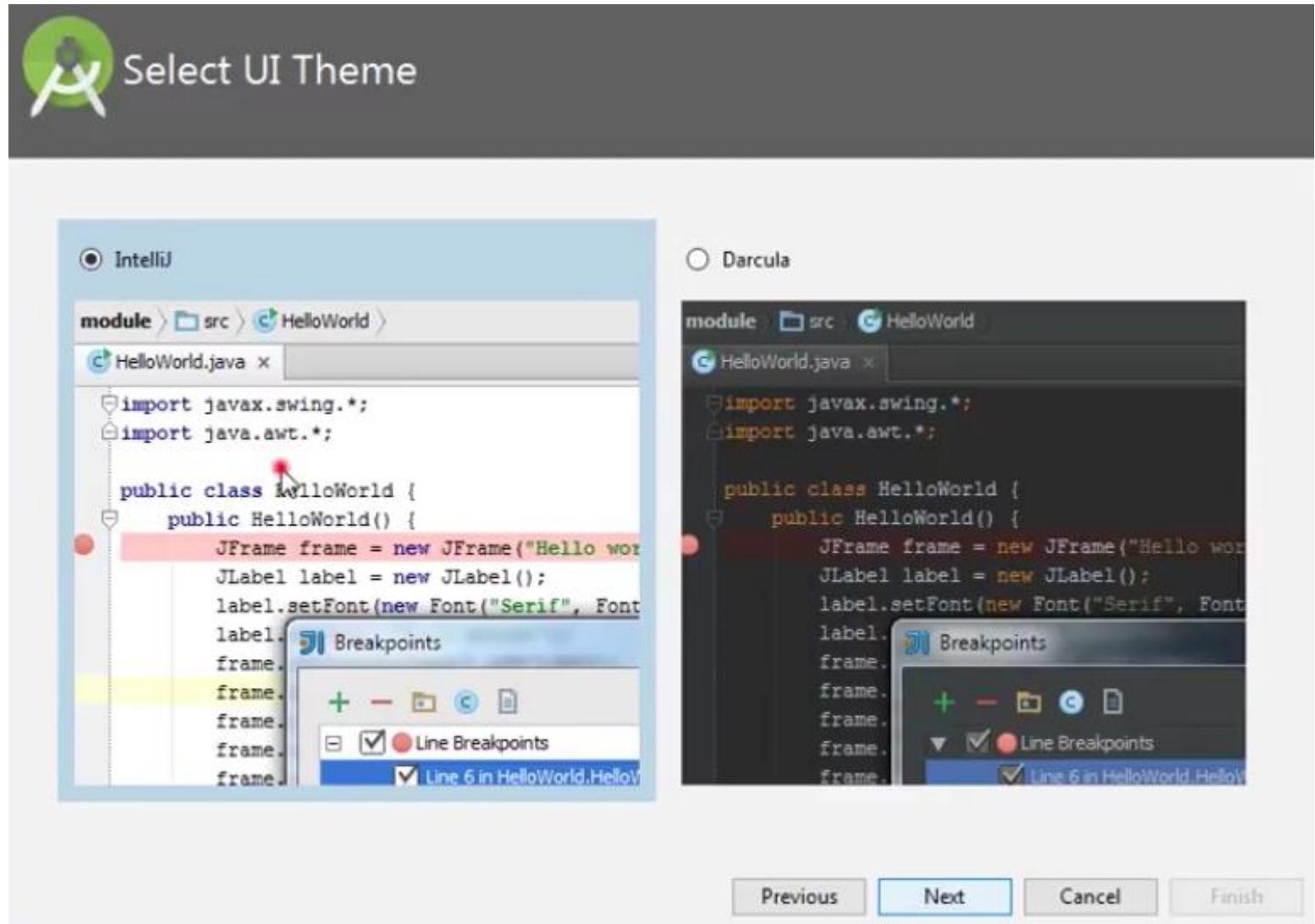
चरण 7: SDK
components को
खोजने के बाद, यह
Welcome dialog box
पर redirect करेगा।



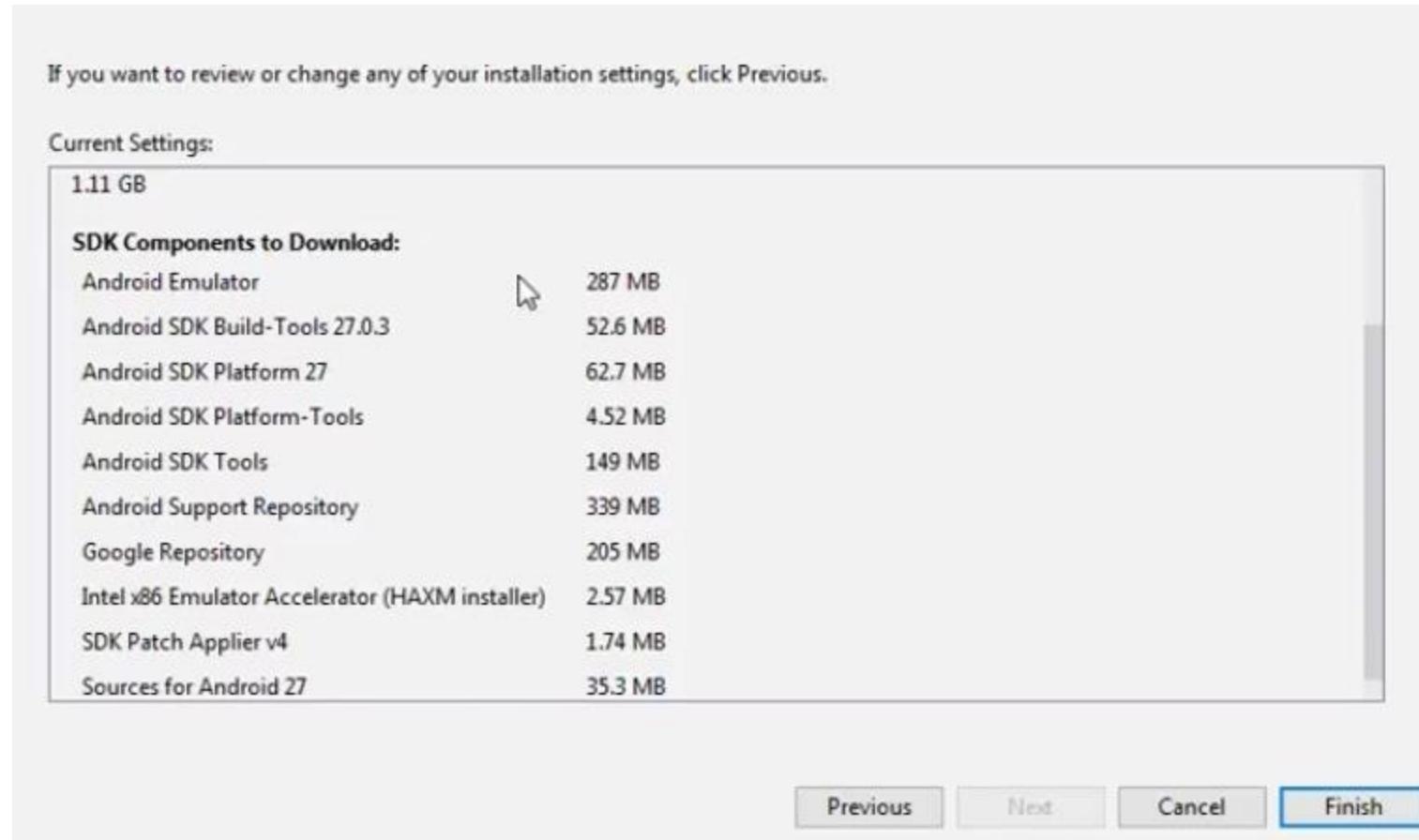
Click on Next



अब थीम चुनें,
चाहे लाइट थीम हो या डार्क ।
लाइट थीम को इंटेलीज थीम
कहा जाता है जबकि डार्क थीम
को ड्रैकुला कहा जाता है ।
आवश्यकतानुसार चुनें।



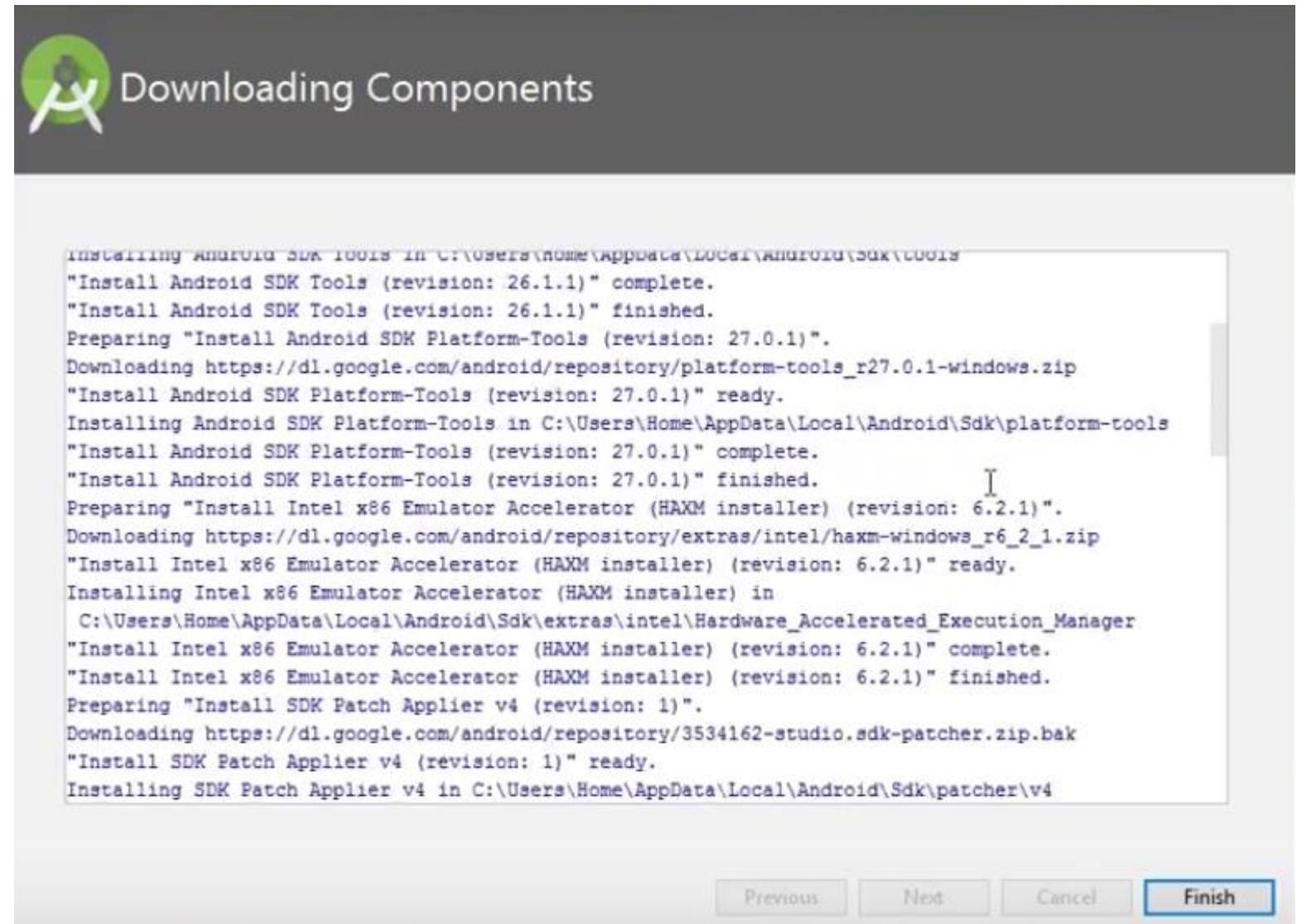
चरण 8: अब SDK components को डाउनलोड करने का समय है।



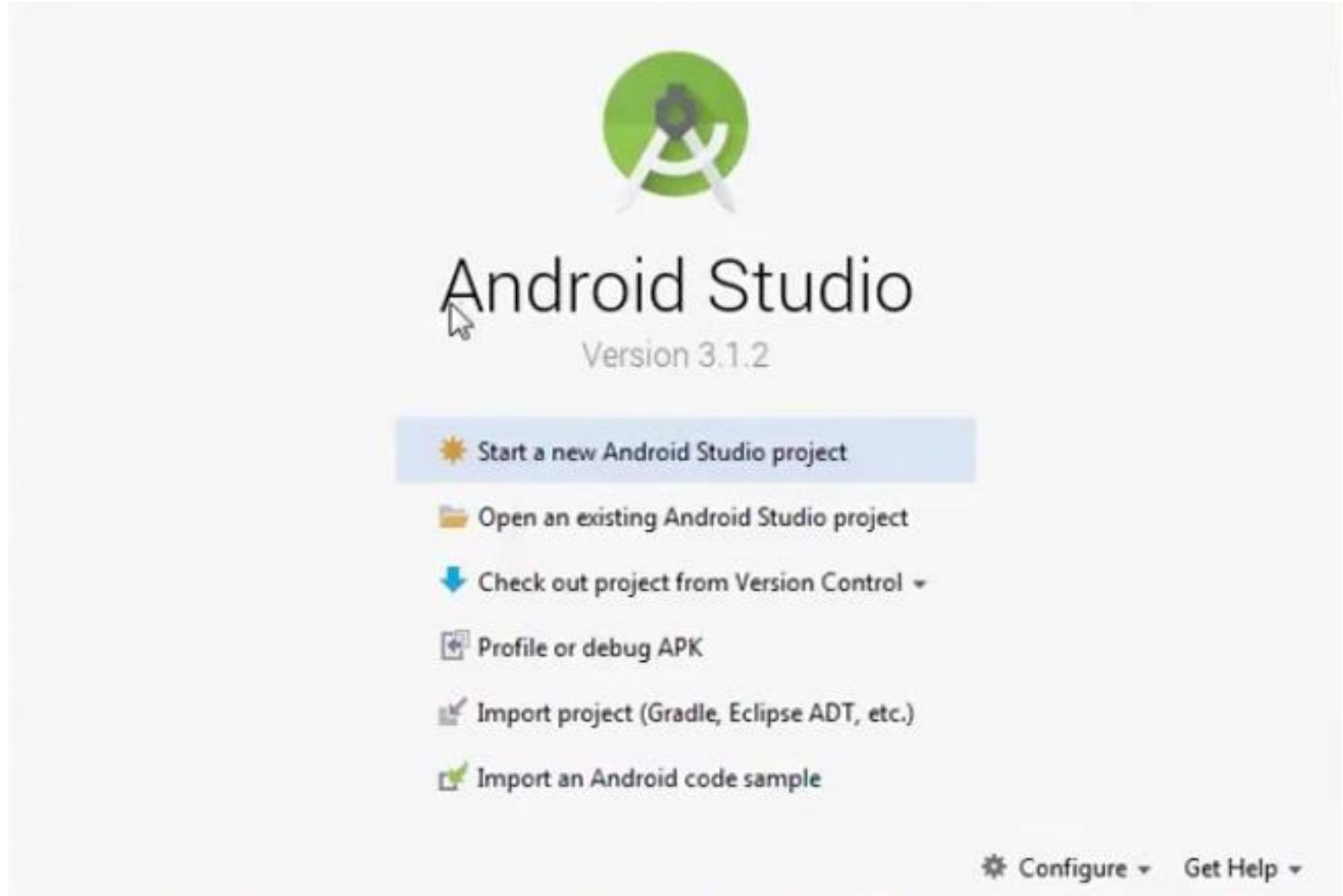
Click on Finish. Components begin to download let it complete.

Android Studio

सफलतापूर्वक कॉन्फिगर हो गया है। अब ऐप लॉन्च करने और बनाने का समय है। इसे लॉन्च करने के लिए फ़िनिश बटन पर क्लिक करें।



चरण 9: नया ऐप बनाने के लिए नया एंड्रॉइड स्टूडियो प्रोजेक्ट प्रारंभ करें पर क्लिक करें।

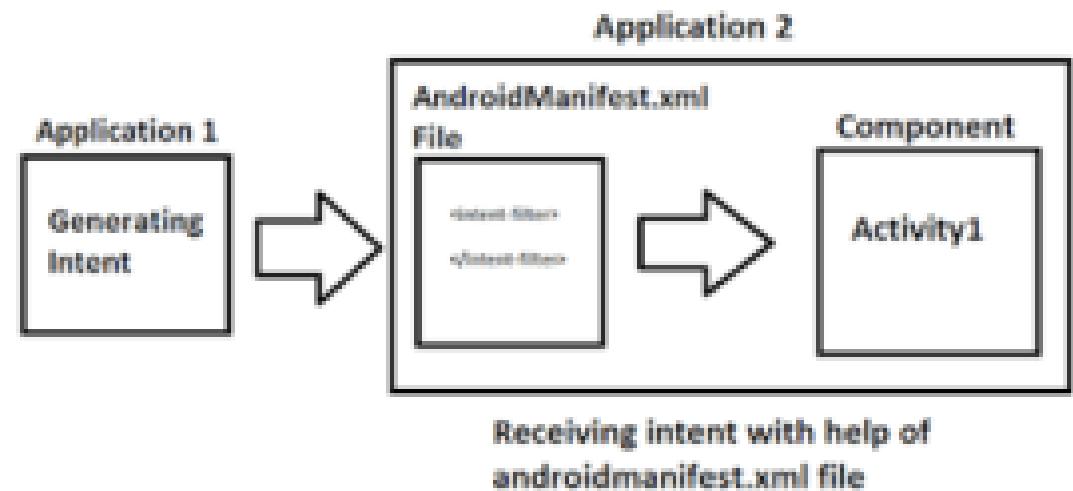


AndroidManifest.xml File

AndroidManifest.xml file में आपके package की सारी जानकारी होती है। आप अपनी एप्लीकेशन में जितने भी components इस्तेमाल करते हैं उन सब की जानकारी AndroidManifest.xml file में मौजूद होती है। जैसे servlets में deployment descriptors होते हैं।

उसी प्रकार AndroidManifest.xml file भी आपकी एप्लीकेशन के लिए deployment descriptors की तरह होती है। ये आपकी application को कंट्रोल करती है। AndroidManifest.xml फ़ाइल को देखकर आप बता सकते हैं कि application में कितने और कौन कौन से components इस्तेमाल हुए हैं।

हर एंड्राइड एप्लीकेशन में कम से कम एक Activity, Service, Broadcast receiver या Content Provider अवश्य होता है। जो की ये सभी components tasks को पूरा करने के लिए Intent को create करते हैं या intent filters के द्वारा खुद को intents के लिए register करते हैं। Intent और intent filters दो अलग अलग element हैं जिन्हे AndroidManifest.xml file जोड़ती है।



किसी भी एप्लीकेशन के components को आपकी एप्लीकेशन के सुरक्षित हिस्से को access करने से पहले permission की ज़रूरत होती है। ये परमिशन AndroidManifest.xml file में declare होती है।

Permission डिक्लेअर करने के लिए <uses-permission> tag का इस्तेमाल किया जाता है।

एप्लीकेशन को create करते समय आप कौन-सा android API इस्तेमाल करते हैं ये आप AndroidManifest.xml file में डिक्लेअर करते हैं।

AndroidManifest.xml File के elements

AndroidManifest.xml file कुछ elements से मिलकर बनी होती है।

<manifest>

<manifest> एलिमेंट AndroidManifest.xml file का root element होता है। इसमें package attribute होता है जो activity क्लास के package name को describe करता है।

AndroidManifest.xml File के elements

<application>

<application> एलिमेंट <manifest> एलिमेंट का sub-element होता है। इस एलिमेंट के कई sub elements होते हैं। इस attribute में commonly icon, label और theme attributes का इस्तेमाल किया जाता है। आइये इनके बारे में कुछ जान लेते हैं।

android:icon – ये आपकी एप्लीकेशन के icon को प्रस्तुत करता है।

android:label – ये आपकी एप्लीकेशन के label को प्रस्तुत करता है।

android:theme – ये आपकी एप्लीकेशन की theme को प्रस्तुत करता है।

<activity>

<activity> एलिमेंट <application> एलिमेंट का sub-element होता है। ये एक activity को दर्शाता है। इस एलिमेंट के 2 प्रमुख attributes होते हैं।

android:name – ये activity class के नाम को प्रस्तुत करता है।

android:label – ये एक label होता है जो स्क्रीन पर डिस्प्ले होता है।

<intent-filter>

<intent-filter> एलिमेंट <activity> एलिमेंट का sub-element होता है। इसके तीन प्रमुख sub-elements होते हैं।

<action> – intent-filter किस प्रकार के action के intent को respond करेगा ये इसमें डिफाइन किया जाता है

<data> – target component का URI इस एलिमेंट में डिफाइन किया जाता है।

<category> – ये activity की category को डिफाइन करता है।

Gradle

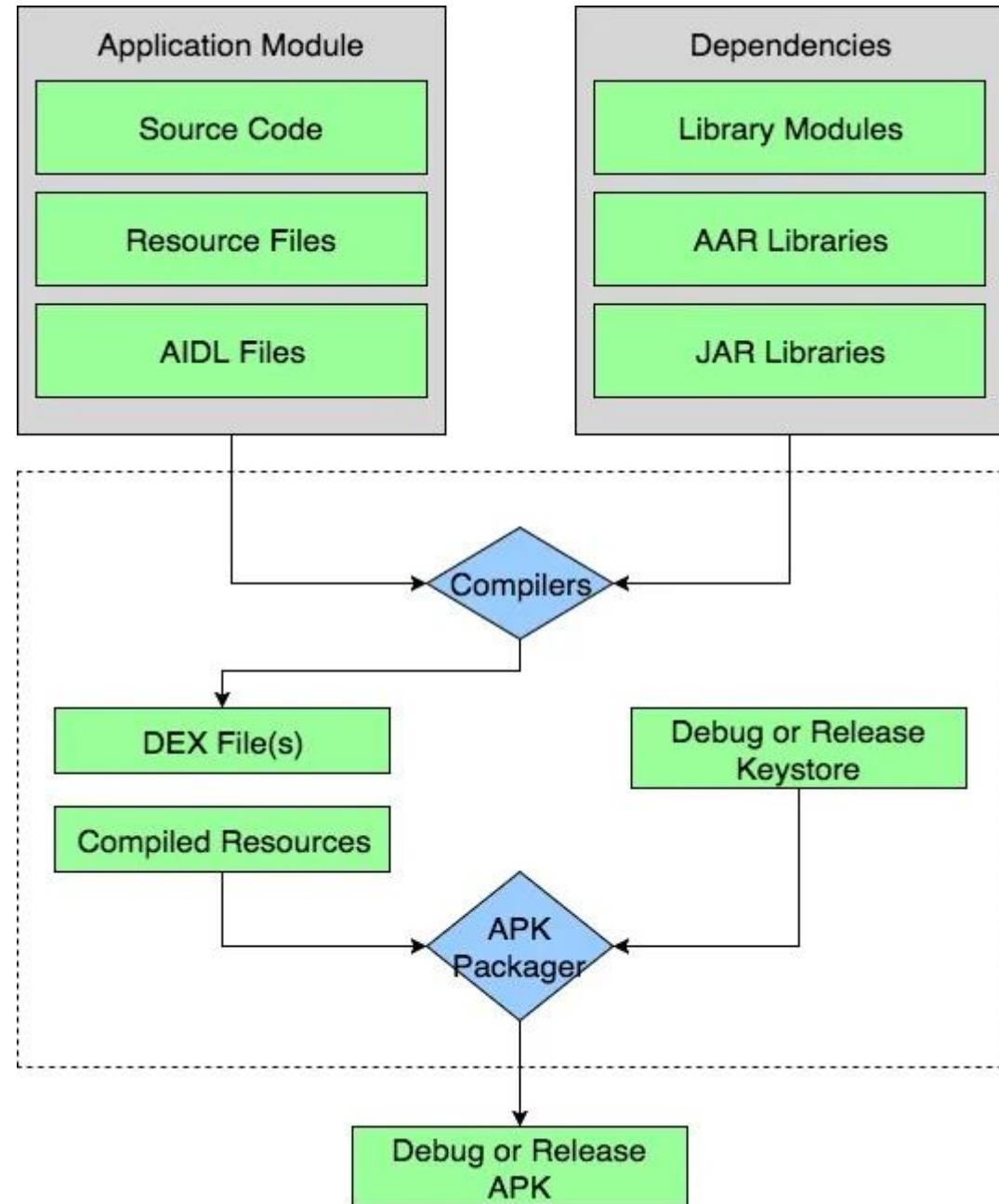
Gradle एक एडवांस्ड बिल्ड सिस्टम है। ये किसी एप्लीकेशन को बिल्ड (पैकेज) करने की प्रोसेस में इस्तेमाल होता है। इससे हम कोड और रिसोर्सेज को आसानी से reuse कर सकते हैं। इससे एक एप्लीकेशन के कई सारे versions आसानी से create किये जा सकते हैं। इससे बिल्ड प्रोसेस को customize करना आसान हो जाता है। Gradle आसानी से IDE के साथ integrate हो जाता है।

Gradle एक बिल्ड सिस्टम है, जो code compilation, testing, deployment and conversion को .dex फाइलों में रूपान्तरित करने तथा डिवाइस पर ऐप चलाने के लिए जिम्मेदार है।

चूंकि एंड्रॉइड स्टूडियो में ग्रैडल सिस्टम पहले से इंस्टॉल आता है, इसलिए हमारे प्रोजेक्ट को बनाने के लिए अतिरिक्त रनटाइम सॉफ्टवेयर इंस्टॉल करने की आवश्यकता नहीं है। जब भी आप एंड्रॉइड स्टूडियो में रन बटन पर क्लिक करते हैं, तो एक ग्रैडल टास्क अपने आप ट्रिगर हो जाता है और प्रोजेक्ट बनाना शुरू कर देता है और ग्रैडल द्वारा अपना टास्क पूरा करने के बाद, ऐप AVD या कनेक्टेड डिवाइस में चलना शुरू हो जाता है।

प्रत्येक एंड्रॉइड स्टूडियो प्रोजेक्ट के लिए
दो फाइलें हैं **build.gradle**, जिनमें से
एक एप्लिकेशन के लिए है और
दूसरी प्रोजेक्ट स्तर (मॉड्यूल स्तर) बिल्ड
फाइलों के लिए है।

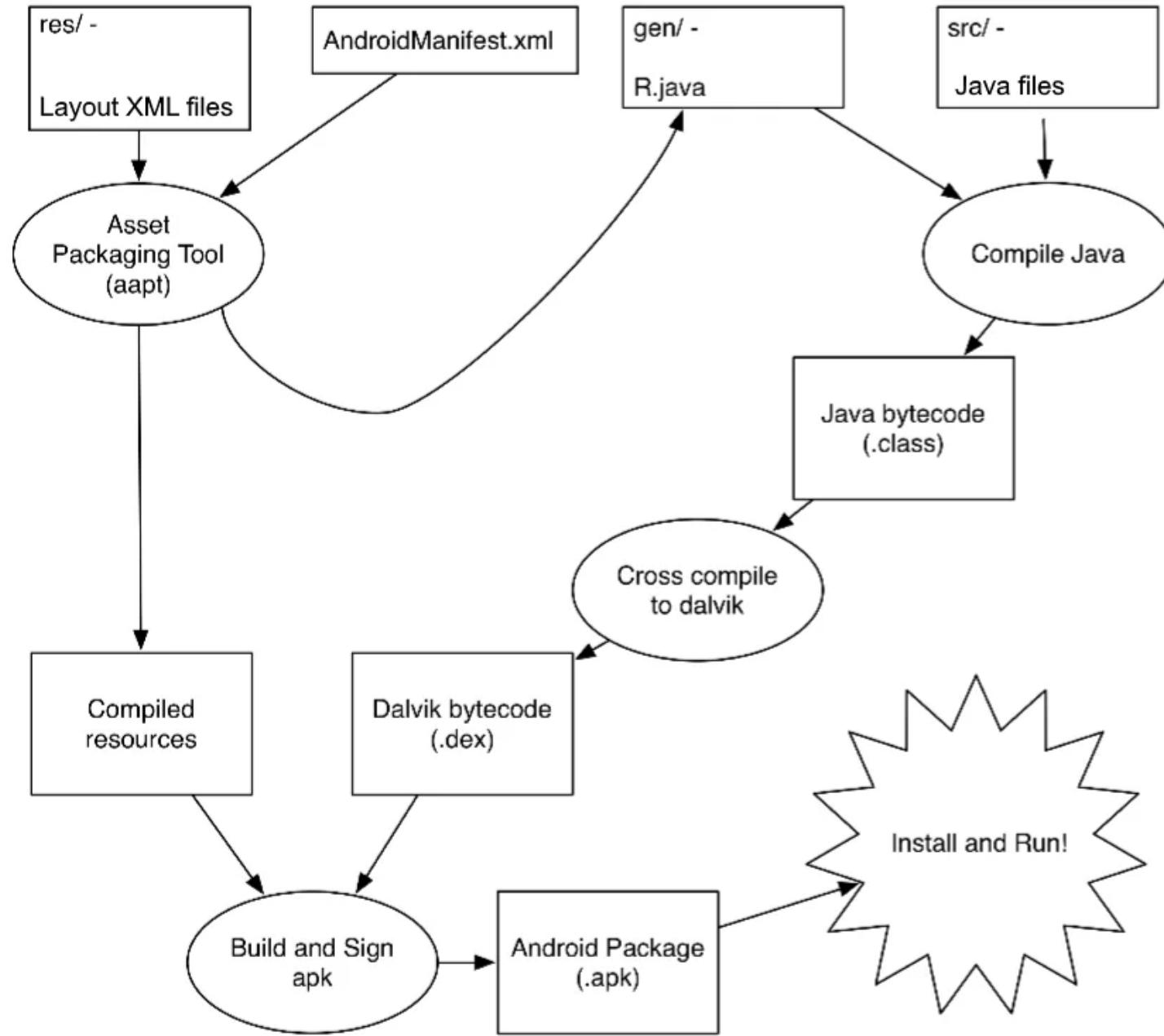
निर्माण प्रक्रिया नीचे दिए गए चित्र के
अनुसार काम करती है।



बिल्ड प्रक्रिया में, कंपाइलर सोर्स कोड, संसाधन, बाहरी लाइब्रेरी JAR फ़ाइलें और **AndroidManifest.xml** (जिसमें एप्लिकेशन के बारे में मेटा-डेटा होता है) लेता है और उन्हें **.dex**(Dalvik Executable फ़ाइलें) फ़ाइलों में परिवर्तित करता है, जिसमें बाइटकोड शामिल होता है। वह बाइटकोड आपके ऐप को चलाने के लिए सभी एंड्रॉइड डिवाइस द्वारा समर्थित है। फिर **APK प्रबंधक****.dex** फ़ाइलों और अन्य सभी संसाधनों को एकल apk फ़ाइल में जोड़ता है।

आप अपने ग्रेडल सिस्टम को कमांड लाइन टूल के माध्यम से भी शुरू कर सकते हैं। इसके लिए निम्नलिखित कमांड का उपयोग किया जाता है:

- `./gradlew build` - (build project)
- `./gradlew clean build` - (build project complete scratch)
- `./gradlew clean build` - (run the test)
- `./gradlew wrapper` - (to see all the available tasks)

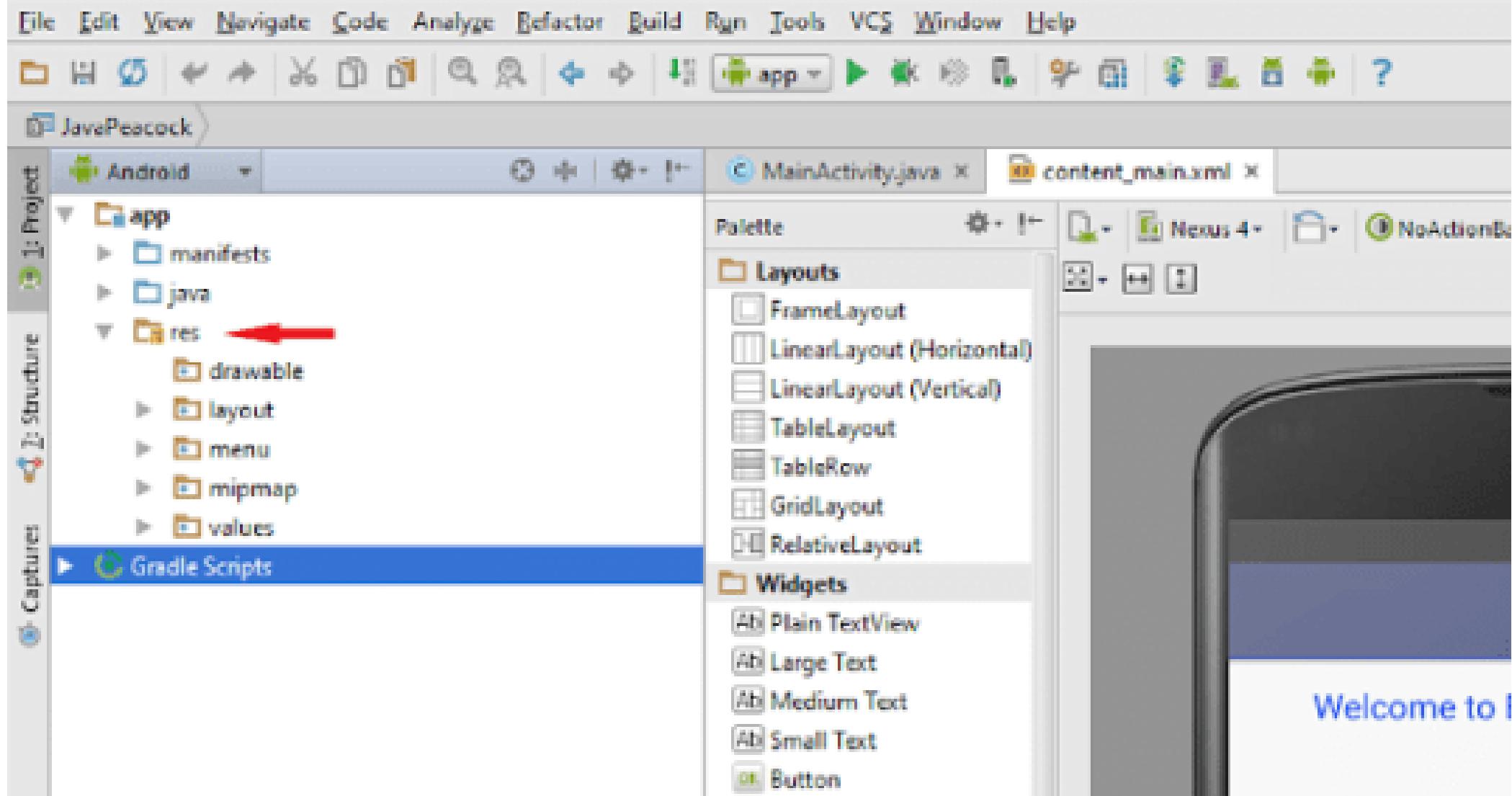


Android Resource

एंड्राइड में, Resources वो files होती हैं जो आपकी एप्लीकेशन के code में शामिल रहती हैं। Resources को एप्लीकेशन के code से अलग रखा जाता है। Resources को एप्लीकेशन से अलग करने से resources को आसानी से maintain किया जा सकता है।

Resources को एंड्राइड एप्लीकेशन directory के **/res folder** में maintain किया जाता है।

JavaPeacock - [C:\Users\navalkishor\AndroidStudioProjects\JavaPeacock] - content_main.xml - Android Studio 1.4



/res फोल्डर की sub directories और उनमें maintain किये जाने वाले resource types की लिस्ट।

Directory	Resource type
/drawable	Images, animations or compiled xml files
/layout	Xml files that defines user interface layout.
/menu	Xml files that defines application menus.
/mipmap	Drawable files for launcher icon
/values	Xml file which contains simple values such as strings, integers and colours.
/animator	Xml files that defines animations.
/colour	Xml files that defines list of colours.
/raw	Arbitrary files to save in their raw form.
/xml	Arbitrary xml files that can be read at run time

Sr.No.	Directory & Resource Type
1	anim/ XML फ़ाइलें जो प्रॉपर्टी एनिमेशन को परिभाषित करती हैं। उन्हें res/anim/ फ़ोल्डर में save किया जाता है और R.anim क्लास से एक्सेस किया जाता है।
2	color/ XML फ़ाइलें जो रंगों की एक सूची को परिभाषित करती हैं। उन्हें res/color/ में save किया जाता है और R.color वर्ग से एक्सेस किया जाता है।
3	drawable/ .png, .jpg, .gif या XML फ़ाइलें जैसी छवि फ़ाइलें जिन्हें बिटमैप, स्टेट लिस्ट, शेप, एनीमेशन ड्रॉएबल में संकलित किया जाता है। उन्हें res/drawable/ में save किया जाता है और R.drawable क्लास से एक्सेस किया जाता है।
4	layout/ XML फ़ाइलें जो उपयोगकर्ता इंटरफ़ेस लेआउट को परिभाषित करती हैं। उन्हें res/layout/ में सहेजा जाता है और R.layout क्लास से एक्सेस किया जाता है।

Sr.No.	Directory & Resource Type
5	<p>menu/</p> <p>XML फाइलें जो अनुप्रयोग मेनू को define करती हैं, जैसे कि Options Menu, Context Menu, or Sub Menu। उन्हें res/menu/ में save किया जाता है और R.menu वर्ग से एक्सेस किया जाता है।</p>
6	<p>raw/</p> <p>Arbitrary files to save in their raw form. You need to call <i>Resources.openRawResource()</i> with the resource ID, which is R.raw.filename to open such raw files.</p>

Sr.No.	Directory & Resource Type
7	<p>values/</p> <p>XML फाइलें जिनमें simple values होते हैं, strings, integers, and colors.</p> <p>For example-</p> <p>arrays.xml for resource arrays, and accessed from the R.array class.</p> <p>integers.xml for resource integers, and accessed from the R.integer class.</p> <p>bools.xml for resource boolean, and accessed from the R.bool class.</p> <p>colors.xml for color values, and accessed from the R.color class.</p> <p>dimens.xml for dimension values, and accessed from the R.dimen class.</p> <p>strings.xml for string values, and accessed from the R.string class.</p> <p>styles.xml for styles, and accessed from the R.style class.</p>

Sr.No.	Directory & Resource Type
8	मनमाने ढंग से XML फ़ाइलें जिन्हें Resources.getXML() को कॉल करके रनटाइम पर पढ़ा जा सकता है। आप यहाँ विभिन्न कॉन्फिगरेशन फ़ाइलों को सहेज सकते हैं जिनका उपयोग रनटाइम पर किया जाएगा।

Alternative Resources

अलग अलग devices की configuration भी अलग अलग होती है। जैसे किसी tablet device की स्क्रीन साइज़ mobile device से बड़ी होती है। इसलिए हर application को अलग अलग device configuration को सपोर्ट करने के लिए alternative resources प्रदान करने चाहिए।

जब आपकी एप्लीकेशन run होती है तो एंड्राइड डिवाइस configuration को detect कर लेता है और उस device के अनुसार resources को load कर देता है। Alternative resources डिफाइन करने के लिए आपको /res फोल्डर में <resources-name>-<config_qualifier> directory क्रिएट करनी होगी।

Android R.java

Android R.java *aapt (Android Asset Packaging Tool)* द्वारा auto-generated फ़ाइल है , जिसमें res/ directory के सभी resources के लिए resource IDs शामिल हैं।

यदि आप activity_main.xml फ़ाइल में कोई component बनाते हैं, तो संबंधित component के लिए आईडी automatically इस फ़ाइल में बनाई जाती है। component पर कोई भी कार्रवाई करने के लिए activity source file में इस आईडी का उपयोग किया जा सकता है।

नोट: यदि आप R.jar फ़ाइल हटाते हैं, तो एंड्रॉयड इसे स्वचालित रूप से बना लेता है।

एंड्रॉइड एमुलेटर

एंड्रॉइड एमुलेटर ऐसे सॉफ्टवेयर हैं जिन्हें विकास और परीक्षण उद्देश्यों के लिए एंड्रॉइड मोबाइल डिवाइस के हार्डवेयर और सॉफ्टवेयर की नकल करने के लिए डिज़ाइन किया गया है। एंड्रॉइड एमुलेटर आपके मैक या पीसी पर चलेंगे ताकि आप अपने डेस्क पर एंड्रॉइड-कंट्रिट एप्लिकेशन बना सकें।

एंड्रॉइड वर्चुअल डिवाइस क्या है?

एंड्रॉयड वर्चुअल डिवाइस मूल रूप से एक डिवाइस कॉन्फ़िगरेशन है। यह एंड्रॉयड मोबाइल, टैबलेट या किसी अन्य एंड्रॉयड डिवाइस के समान है। डेवलपर्स अपने द्वारा विकसित किए गए एप्लिकेशन को AVD पर टेस्ट कर सकते हैं। सरल तरीके से समझने के लिए AVD एक ऐसा सॉफ्टवेयर है जो वास्तविक हार्डवेयर एंड्रॉयड डिवाइस की नकल करता है। इसमें हार्डवेयर इमेज, स्टोरेज सिस्टम, एप्लिकेशन और अन्य चीजें होती हैं, बिल्कुल असली डिवाइस की तरह।

एंड्रॉइड वर्चुअल डिवाइस क्यों?

हमें किसी Android प्रोजेक्ट या एप्लिकेशन के विकास के दौरान इसकी आवश्यकता होती है। चूंकि यह किसी एप्लिकेशन का वर्चुअल प्रतिनिधित्व प्रदान करता है, इसलिए यह हमें अपने एप्लिकेशन को उसी पर चलाने में मदद करता है। यह सुनिश्चित करने के लिए आवश्यक है कि हमारा ऐप सही तरीके से काम करे। यह वास्तविक डिवाइस की तुलना में बेहतर है क्योंकि यह मोबाइल डिवाइस को कनेक्ट करने और बार-बार एप्लिकेशन इंस्टॉल करने के कार्य को समाप्त करता है। यह हमें ऐप और उसके कार्यों को सही ढंग से परखने और किसी भी तरह की खराबी को बिना किसी देरी के तुरंत ठीक करने में मदद करता है।

Logcat

लॉगकैट विंडो वह जगह है जहाँ किसी एप्लिकेशन के चलने पर विभिन्न संदेश प्रिंट किए जा सकते हैं। मान लीजिए, आप अपना एप्लिकेशन चला रहे हैं और दुर्भाग्य से प्रोग्राम क्रैश हो जाता है। तब, लॉगकैट विंडो आपके एमुलेटर द्वारा फेंके गए सभी संदेशों को एकत्रित करके और देखकर आउटपुट को डीबग करने में आपकी मदद करने जा रही है। इसलिए, यह ऐप डेवलपमेंट के लिए एक बहुत ही उपयोगी घटक है क्योंकि यह लॉगकैट बहुत सारे सिस्टम संदेशों को डंप करता है और ये संदेश वास्तव में एमुलेटर द्वारा फेंके जाते हैं।

इसका मतलब यह है कि जब आप अपने एमुलेटर में अपना ऐप चलाते हैं, तो आपको बहुत सारे संदेश दिखाई देंगे, जिसमें सभी जानकारी, सभी वर्बाज़ संदेश और आपके एप्लिकेशन में आने वाली सभी त्रुटियाँ शामिल होंगी।

लॉग क्लास में बहुत सारी विधियाँ मौजूद हैं:

v(String, String)	verbose
d(String, String)	debug
i(String, String)	information
w(String, String)	warning
e(String, String)	error

Method	The output will be printed in
verbose	black
debug	blue
information	green
warning	red
error	orange

```
// for verbose Log.v("TAG", "MESSAGE");
// for debug Log.d("TAG", "MESSAGE");
// for information Log.i("TAG", "MESSAGE");
// for warning Log.w("TAG", "MESSAGE");
// for error Log.e("TAG", "MESSAGE");
```

For Example, A verbose log message can be written as:

Log.v("MainActivity", "We are under the Main Activity");

Android UI Layouts

एंड्रॉइड लेआउट का उपयोग यूजर इंटरफ़ेस को define करने के लिए किया जाता है जो UI controls या widgets रखता है जो एंड्रॉइड एप्लिकेशन या activity screen की स्क्रीन पर दिखाई देगा। आम तौर पर, हर एप्लिकेशन व्यू और व्यूग्रुप का combination होता है। जैसा कि हम जानते हैं, एक एंड्रॉइड एप्लिकेशन में बड़ी संख्या में गतिविधियाँ होती हैं और हम कह सकते हैं कि प्रत्येक गतिविधि एप्लिकेशन का एक पेज है। इसलिए, प्रत्येक गतिविधि में कई यूजर इंटरफ़ेस components होते हैं और वे components व्यू और व्यूग्रुप के उदाहरण होते हैं। लेआउट में सभी तत्व व्यू और व्यूग्रुप ऑब्जेक्ट के hierarchy का उपयोग करके बनाए जाते हैं।

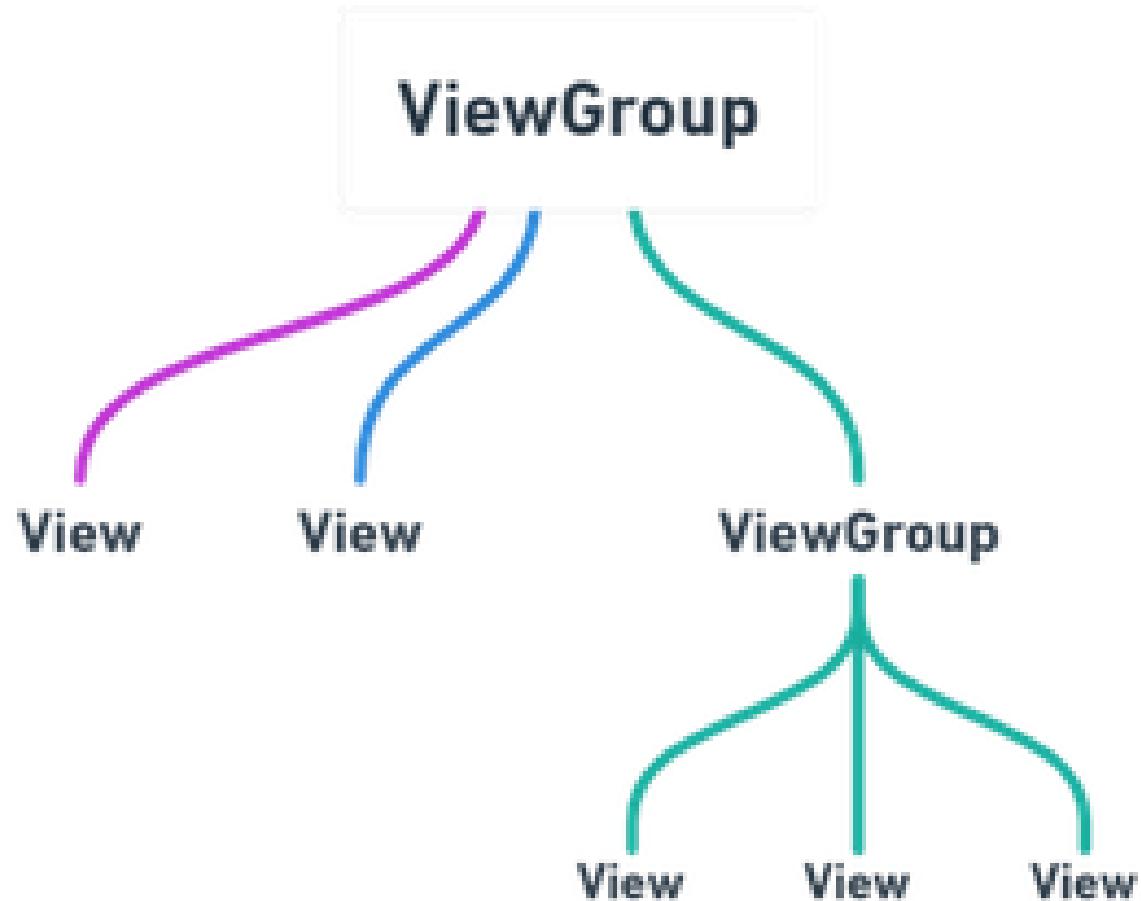
View

व्यू को यूजर इंटरफ़ेस के रूप में परिभाषित किया जाता है जिसका उपयोग इंटरैक्टिव UI component

जैसे [TextView](#) , [ImageView](#) , [EditText](#) , [RadioButton](#) आदि बनाने के लिए किया जाता है, और यह इवेंट हैंडलिंग और ड्राइंग के लिए जिम्मेदार होता है। इन्हें आम तौर पर विजेट कहा जाता है।

एक व्यूग्रुप लेआउट और लेआउट पैरामीटर के लिए एक बेस क्लास के रूप में कार्य करता है जो अन्य व्यू या व्यूग्रुप को होल्ड करता है और लेआउट गुणों को परिभाषित करता है। इन्हें आम तौर पर लेआउट कहा जाता है।

Types of Android Views



- EditText
- ProgressBar
- Button
- Spinner
- ImageButton
- TimePicker
- ToggleButton
- DatePicker
- RadioButton
- SeekBar
- RadioGroup
- AlertDialog
- CheckBox
- Switch
- AutoCompleteTextView
- RatingBar

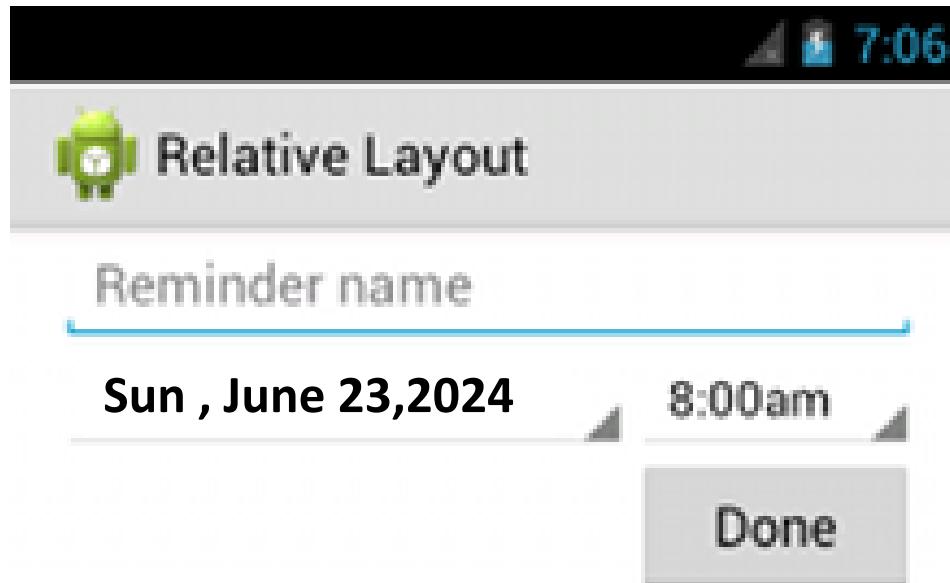
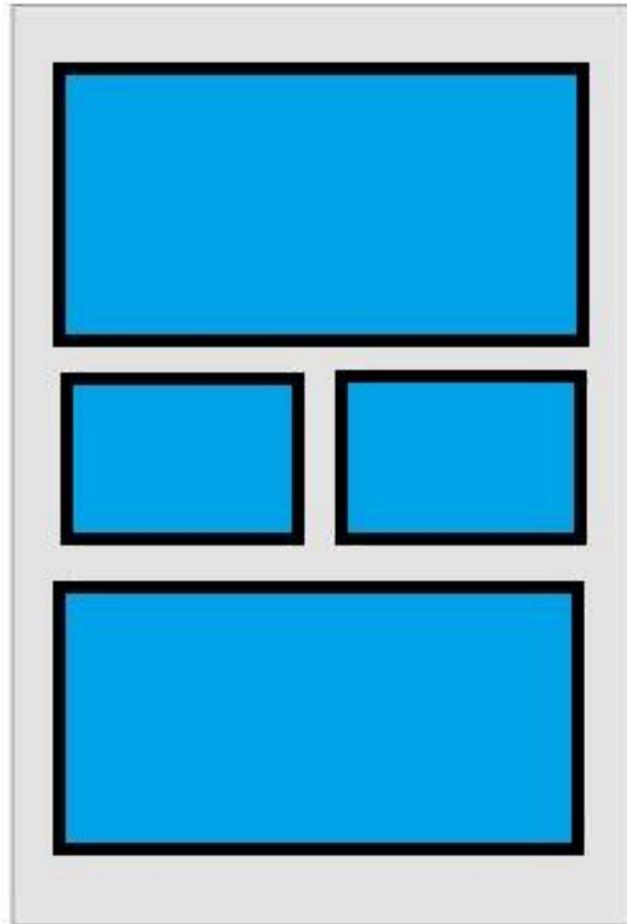
Types of Android Layout

- Android Linear Layout
- Android Relative Layout
- Android Constraint Layout
- Android Frame Layout
- Android Table Layout
- Android Web View
- Android List View
- Android Grid View

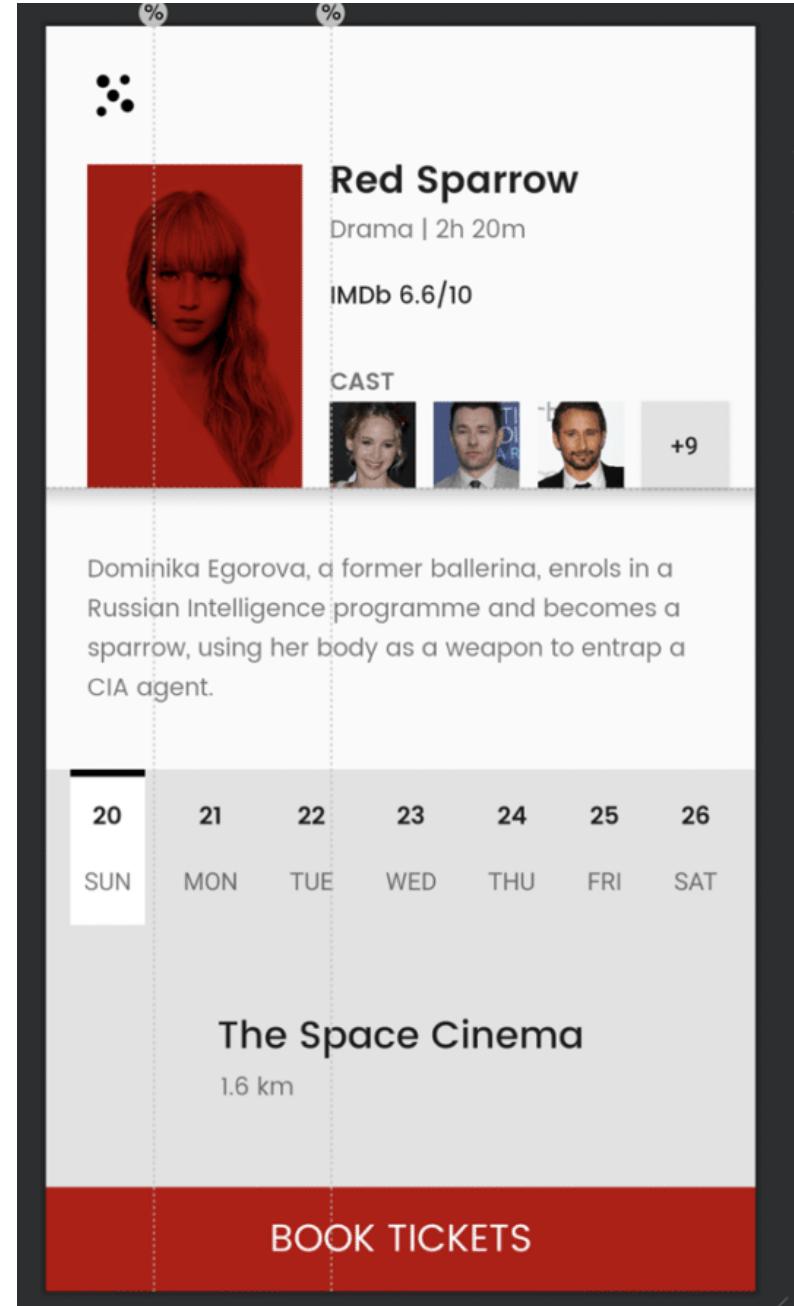
•एंड्रॉइड लीनियर लेआउट: लीनियर लेआउट एक व्यूग्रुप उपवर्ग है, जिसका उपयोग ओरिएंटेशन प्रॉपर्टी के आधार पर चाइल्ड व्यू तत्वों को एक-एक करके किसी विशेष दिशा में क्षैतिज या लंबवत रूप से प्रदान करने के लिए किया जाता है।



• एंड्रॉइड रिलेटिव लेआउट: RelativeLayout एक ViewGroup उपवर्ग है, जिसका उपयोग चाइल्ड व्यू तत्वों की एक दूसरे के सापेक्ष स्थिति को निर्दिष्ट करने के लिए किया जाता है (जैसे A को B के दाईं ओर) या पैरेंट के सापेक्ष (पैरेंट के शीर्ष पर स्थिर)।



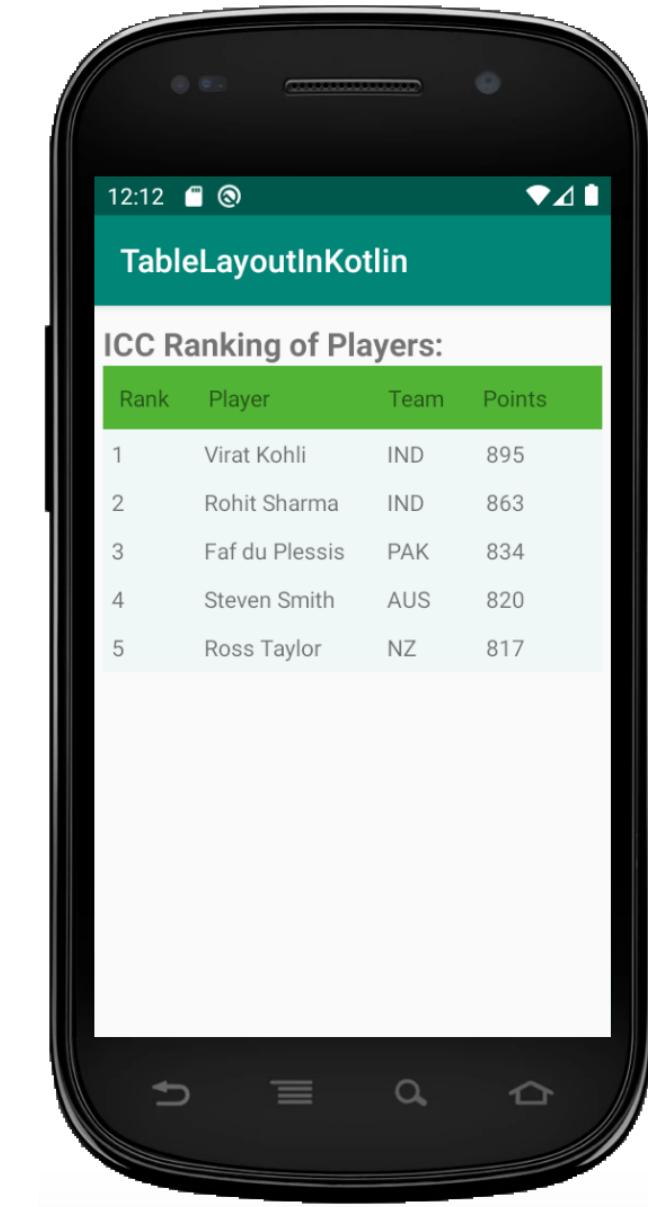
•**एंड्रॉड कंस्ट्रेन्ट लेआउट:** कंस्ट्रेन्ट लेआउट एक व्यूग्रुप सबकलास है, जिसका उपयोग हर चाइल्ड व्यू के लिए लेआउट कंस्ट्रेन्ट की स्थिति को अन्य मौजूद व्यू के सापेक्ष निर्दिष्ट करने के लिए किया जाता है। एक कंस्ट्रेन्ट लेआउट एक रिलेटिव लेआउट के समान है, लेकिन इसमें अधिक शक्ति होती है।



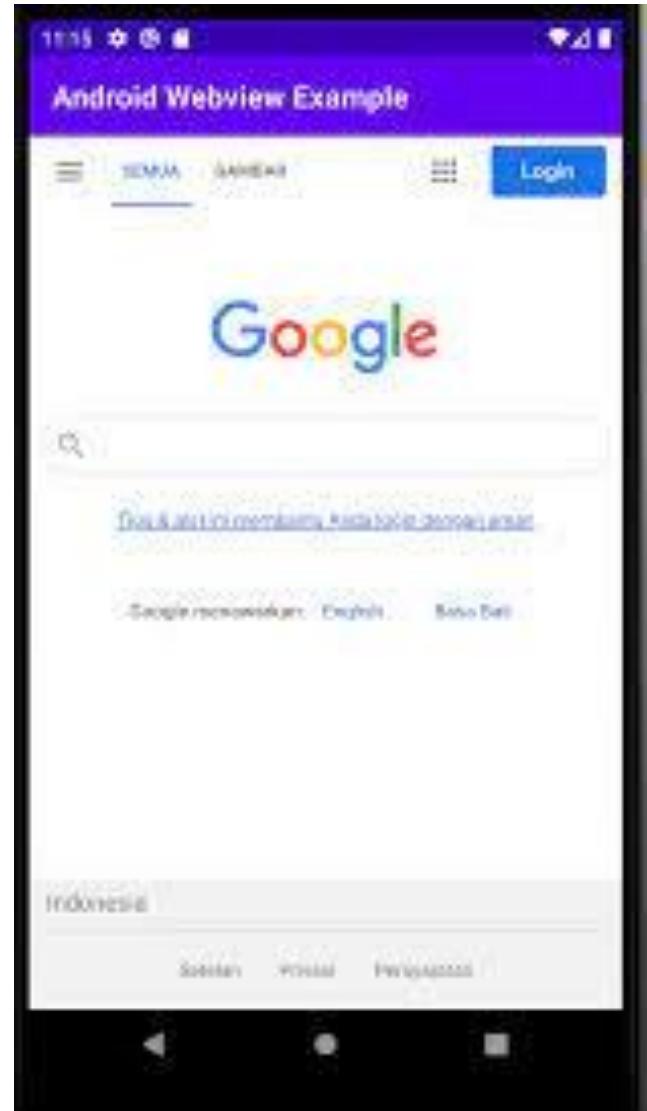
•एंड्रॉइड फ्रेम लेआउट: फ्रेमलेआउट एक व्यूग्रुप उपवर्ग है, जिसका उपयोग फ्रेमलेआउट के अंदर केवल एक ही दृश्य प्रदर्शित करने के लिए इसमें शामिल व्यू तत्वों की एक दूसरे के शीर्ष पर स्थिति निर्दिष्ट करने के लिए किया जाता है।



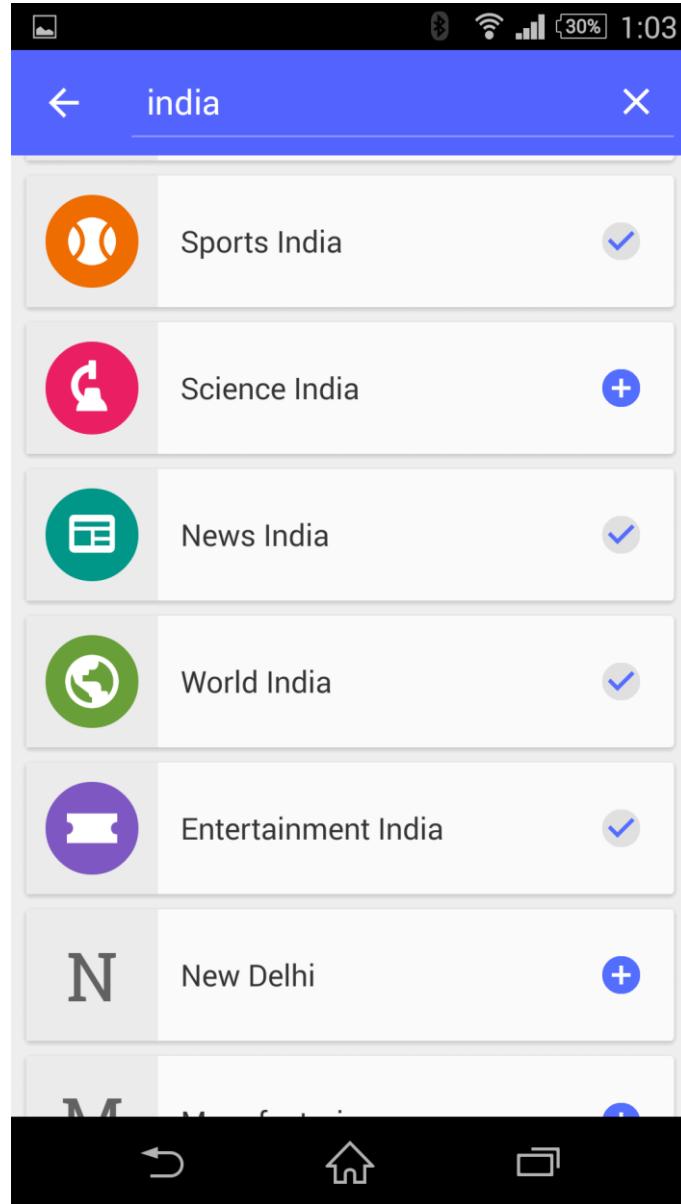
•**एंड्रॉइड टेबल लेआउट:** टेबललेआउट एक व्यूग्रुप उपवर्ग है, जिसका उपयोग पंक्तियों और स्तंभों में चाइल्ड व्यू तत्वों को प्रदर्शित करने के लिए किया जाता है।



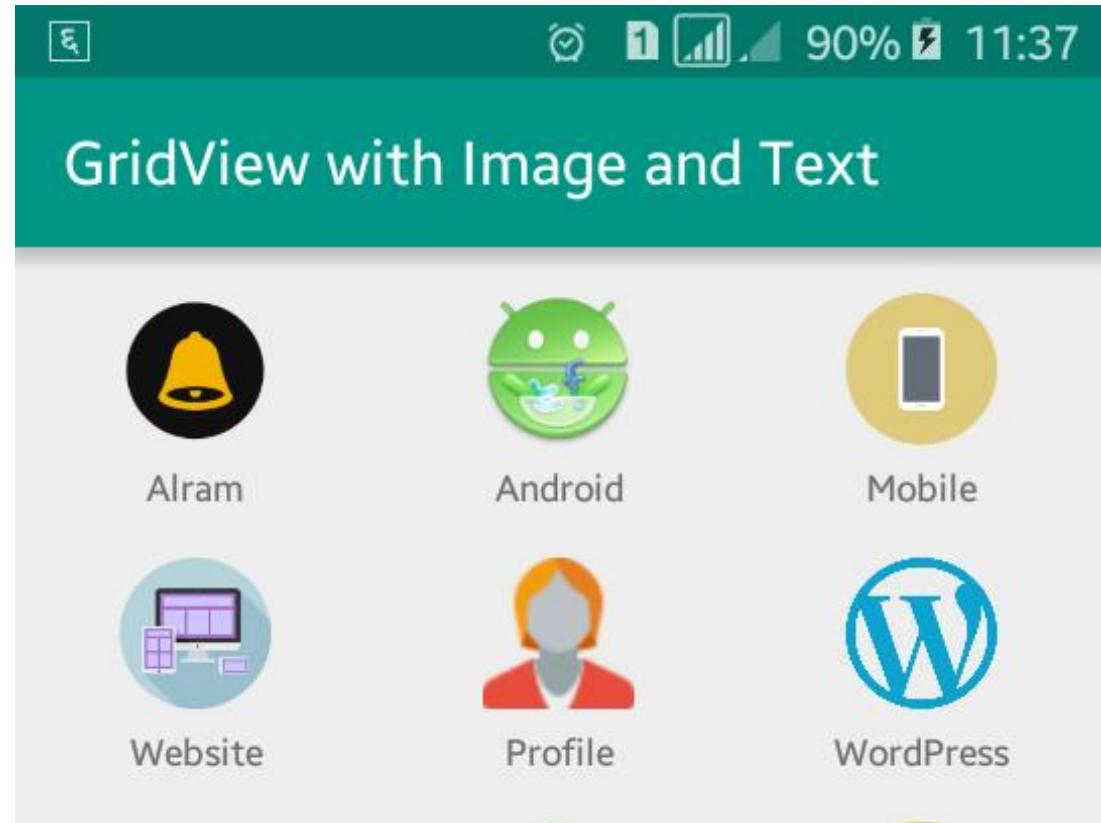
•**एंड्रॉइड वेब व्यूः** वेबव्यू एक ब्राउज़र है जिसका उपयोग हमारे गतिविधि लेआउट में वेब पेजों को प्रदर्शित करने के लिए किया जाता है।



•**एंड्रॉयड सूची दृश्य:** सूची दृश्य एक दृश्य समूह है, जिसका उपयोग एकल कॉलम में आइटमों की स्क्रॉल करने योग्य सूची प्रदर्शित करने के लिए किया जाता है।



•**एंड्रॉइड ग्रिड व्यूः** ग्रिड व्यू एक व्यूग्रुप है जिसका उपयोग पंक्तियों और स्तंभों के ग्रिड दृश्य में आइटमों की स्क्रॉल करने योग्य सूची प्रदर्शित करने के लिए किया जाता है।



Android Activity

Android Activity – Activity एक User Interface (UI) Concept होता है जो सामान्यतः किसी Android Application की एक Single Screen को Represent करता है। इस User Interface Screen पर जरूरत के अनुसार एक या अधिक Views हो सकते हैं लेकिन Views का होना Compulsory नहीं होता। यानी एक ऐसा Android App भी हो सकता है कि जिसकी Activity में कोई भी View न हो।
किसी Activity को हम हमारे Android App का एक ऐसा Part मान सकते हैं, जो User को कोई Action Perform करने में मदद करता है। सरल शब्दों में कहें तो Activity, User Interface के रूप में वह माध्यम होता है, जो User Interactions के कारण Perform होने वाले विभिन्न Actions को Handle करता है।

उदाहरण के लिए जब User किसी Android App की किसी Screen पर दिखाई देने वाले किसी **Button** को Tap करता है, तो उस Tapping Action के Response में Perform होने वाले Task को इसी **Activity** Part में Specify व Implement किया जाता है।

मानलो कि हम चाहते हैं कि जब User किसी Button को Click करे, तो वह Android App **Close** हो जाए।

इस जरूरत को पूरा करने के लिए **Button Click Action** के **Response** में Android App को Close करने से सम्बंधित Programming Codes (Program Logics) को **Activity** के अन्तर्गत ही लिखा जाता है, जो कि एक Java Class होती है।

किसी Activity के अन्तर्गत सामान्यतः Data View करना, Data Create करना व Data Edit करने से सम्बंधित Tasks को ही Complete किया जाता है। अन्य शब्दों में कहें तो CRUID (Create, Read, Update, Insert, Delete) Operations को Android App के Activity Class File में ही Perform किया जाता है। Activity किसी भी Android App का एक Compulsory Part होता है और किसी भी Android App को Open करने पर जो सबसे पहला Screen Display होता है, वह Activity ही होता है।

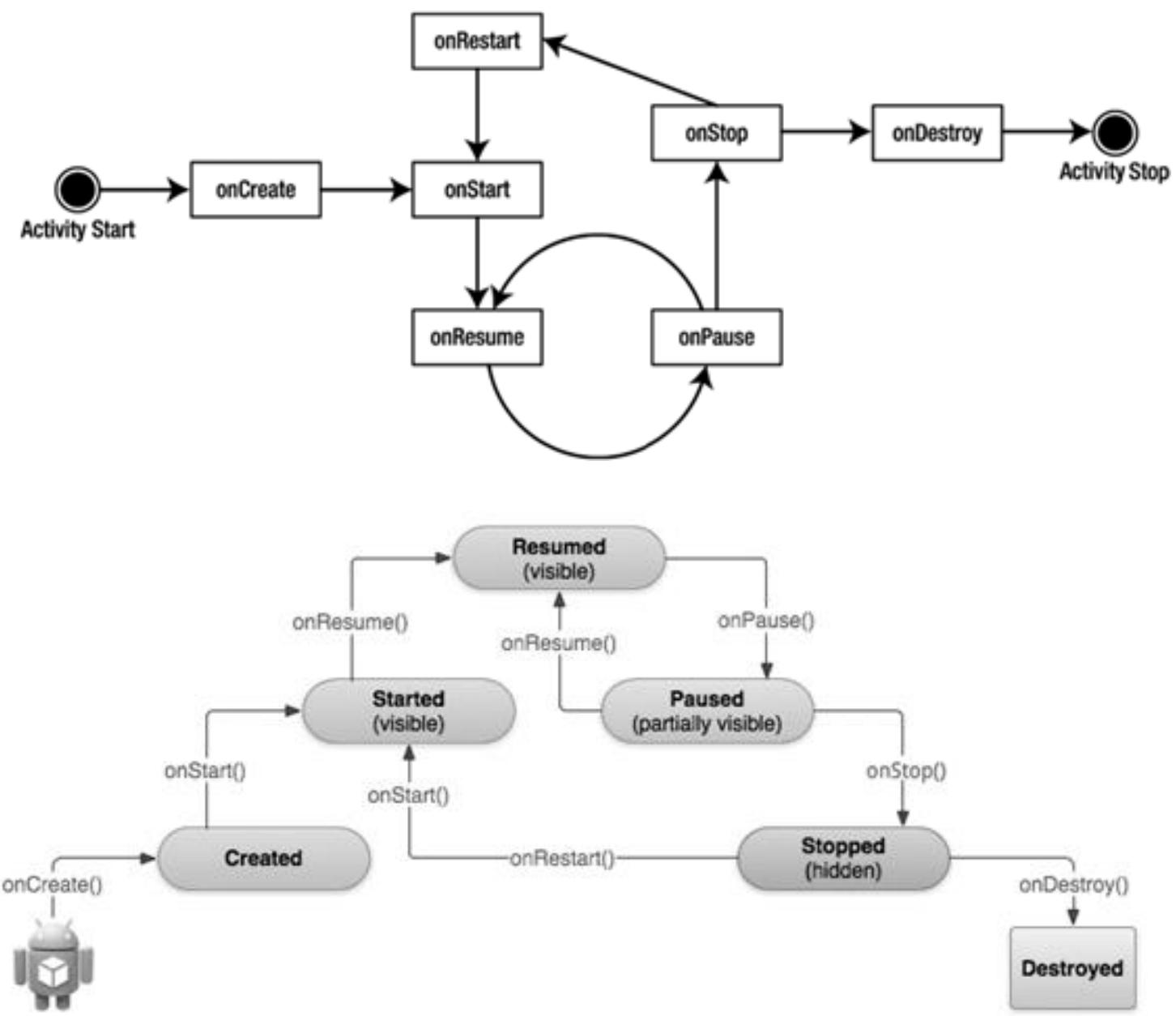
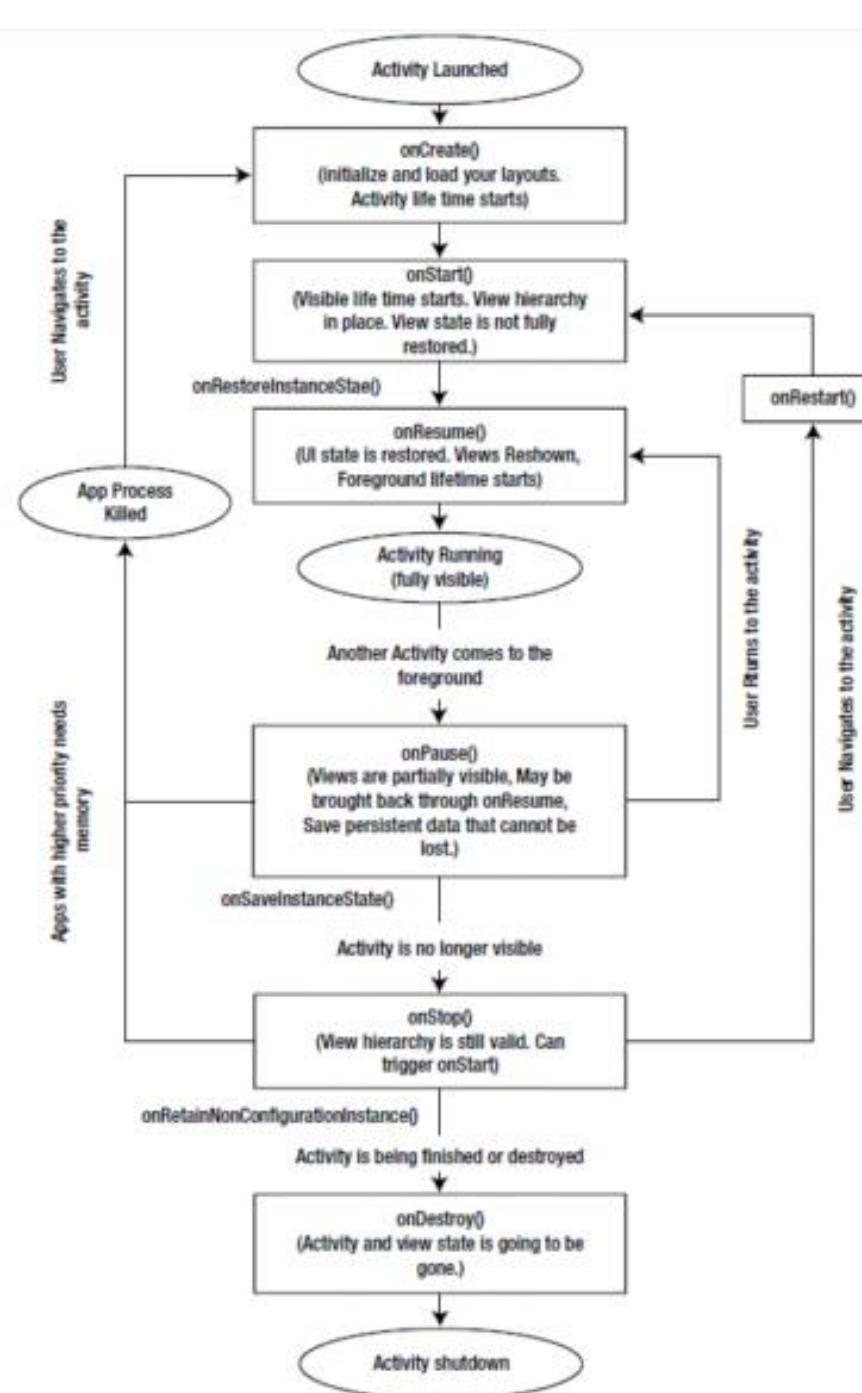
Android App Lifecycle

Android App की एक Lifecycle होती है और इस Lifecycle के विभिन्न Important Points को कुछ निश्चित Methods द्वारा Represent किया जाता है। फलस्वरूप हमें किसी Particular Activity की किस State में क्या Operation Perform करवाना है, उस Operation से सम्बंधित Program Logics को उस Particular State से सम्बंधित Method को Override करके उसी Overridden Method में Specify करना होता है। एक Android App के Lifecycle को User की जरूरत व उपलब्ध Resources के आधार पर पूरी तरह से **Android Operating System** यानी **Android Platform** द्वारा Manage किया जाता है।

उदाहरण के लिए हालांकि User अपने Android Device पर Web Browser Launch करना चाहता है, लेकिन Web Browser App Launch होगा या नहीं, इस बात को Device का Android System तय करता है। इसलिए Android Device का Android Platform ही Ultimate Manager होता है और इस बात को तय करता है कि कोई Specific Task Complete होगा या नहीं और यदि होगा तो किस तरह से।

उदाहरण के लिए मानलो कि एक User A किसी से User B से Phone Call पर बात कर रहा है। User B, User A से कोई Information मांगता है, जो कि उस Webpage में है, जिसका URL User A के Mail Box में Saved एक Email में है। वह Information देने के लिए User A को Phone Call के दौरान ही अपना Email App Open करना पड़ता है और उस Webpage के Hyperlink पर Click करना पड़ता है, जिस पर Information Exist है।

हम समझ सकते हैं कि इस पूरी प्रक्रिया के दौरान User A को कुल चार Android Apps के बीच Switch करना पड़ता है: **Home Application, Talk Application, Email Application व Web Browser Application** और जब User A, एक Application से दूसरे Application पर Navigate करता है, तो इस Navigation में उसे किसी प्रकार की कोई परेशानी पैदा नहीं होती यानी User का Experience काफी Seamless होता है। हालांकि प्रत्येक Application से दूसरे पर Switching के दौरान Background में Android System, Current Application की State को Save व Restore करता रहता है।



1. `onCreate()`: यह गतिविधि शुरू होने पर XML फ़ाइल में दिए गए UI को प्रदर्शित करता है। यह `SetContentView()` विधि को कॉल करता है।

2. `onStart()`: जब हमारी गतिविधि प्रदर्शित होने की प्रक्रिया में होती है तो इस कॉलबैक विधि को कॉल किया जाता है। और इस विधि के बाद `onResume()` कॉलबैक को कॉल किया जाता है।

3. `onResume()`: अनुप्रयोग की मुख्य कार्यक्षमता इस कॉलबैक विधि के भीतर लिखी गई है। जब गतिविधि प्रदर्शित होने वाली होती है तो इस कॉलबैक विधि को कॉल किया जाता है। मान लीजिए कि यदि आप एक नई गतिविधि शुरू करते हैं जो पहले से चल रही गतिविधि को छिपा देती है, तो `onResume()` को तब कॉल किया जाता है जब छिपी हुई गतिविधि फिर से स्क्रीन पर दिखाई देती है।

4. onPause(): जब एप्लिकेशन को रोका जा रहा हो (न्यूनतम किया जा रहा हो) तो आप जिस कोड को निष्पादित करना चाहते हैं उसे इस कॉलबैक विधि में लिखा जाना चाहिए और जब उपयोगकर्ता एप्लिकेशन को फिर से शुरू करता है तो onResume() कॉलबैक विधि को फिर से बुलाया जाएगा।

5. onStop(): जब किसी गतिविधि को न्यूनतम किया जा रहा हो तो onPause() कॉलबैक तुरंत कॉल किया जाता है और कुछ मिलीसेकंड के बाद onStop को कॉल किया जाएगा। onStop() एप्लिकेशन के API कॉल को रोक देगा।

6. onDestroy(): onDestroy कॉलबैक को तब कॉल किया जाता है जब आपको सभी ऑपरेशन बंद करने की आवश्यकता होती है।

Android App Permission

मोबाइल में अप्प परमिशन का मतलब किसी अप्प को अपने मोबाइल की कुछ चीजों जैसे कैमेरा, मीडिया, लोकेशन आदि को एक्सेस करने की परमिशन से होता है, सबसे पहले आपको अपने मोबाइल के डाटा के बारे में पता होना जरूरी है, आप अपने मोबाइल में जो भी फोटोज, वीडियोसज़ डॉक्यूमेंट आदि को स्टोर करते हैं यानी आपके डिवाइस में जो फोटोज, वीडियो, डॉक्यूमेंट आदि स्टोरेज में या मेमोरी कार्ड में सेव होते हैं वही आपका डाटा होता है, और जब भी आप इंटरनेट से किसी App को अपने डिवाइस में डाउनलोड करके उसे फोटो/मीडिया/स्टोरेज को एक्सेस करने की परमिशन दे देते हैं तो आप उस अप्प को अपने मोबाइल डाटा की परमिशन भी देते हैं लगभग सभी एंड्राइड यूजर प्ले स्टोर से ही किसी भी एप्प को डाउनलोड करते होंगे और आप मेसे बहुत से लोग ऐसे होंगे जो ये समझते होंगे कि प्ले स्टोर पर हमें जो Apps मिल रहे हैं

Android App Permission

वो सभी गूगल ने बनाये हैं लेकिन ऐसा बिलकुल भी नहीं है, गूगल प्ले स्टोर पर बिलियंस अप्प्स हैं यानि कोई भी जिसका गूगल डेवलपर अकाउंट है वो अपनी एप्लीकेशन को प्ले स्टोर पर अपलोड कर सकता है। इससिए गूगल प्ले स्टोर से किसी भी एप्प को डाउनलोड करते टाइम आप ये सोचकर उसे डाउनलोड न करें कि इसे गूगल ने बनाया है इसलिए ये पूरी तरह सिक्योर हैं।

PERMISSIONS के प्रकार

१- Install Time permissions

२- Runtime permissions

३- Special permissions

सामान्य अनुमतियाँ

यदि उपयोगकर्ता की गोपनीयता को बहुत कम या कोई खतरा नहीं है, तो अनुमति सामान्य अनुमति श्रेणी में आती है। उदाहरण के लिए, यदि आप डेटा और समय प्राप्त करना चाहते हैं, तो इन चीजों से उपयोगकर्ता की कोई गोपनीयता नहीं जुड़ी होती है, और आपको इस स्थिति में उपयोगकर्ता से दिनांक या समय का उपयोग करने के लिए कहने की आवश्यकता नहीं है। आप AndroidManifest.xml फाइल में अनुमति जोड़कर सीधे इस सुविधा का उपयोग कर सकते हैं।

ब्लूटूथ

वॉलपेपर सेट करो

ब्लूटूथ_एडमिन

कंपन

इंटरनेट

चेतावनी तय करें

Signature Permissions

एंड्रॉइड सिस्टम इंस्टॉलेशन के दौरान ये अधिकार देता है, लेकिन इसमें एक दिक्कत है। अनुमति मांगने वाले ऐप को उसी हस्ताक्षर से हस्ताक्षरित होना चाहिए, जिस हस्ताक्षर से आवश्यक अनुमति को परिभाषित करने वाला ऐप है।

- 1.BIND_ऑटोफिल_सेवा
- 2.BIND_CARRIER_SERVICE
- 3.BIND_TV_INPUT
- 4.BIND_वॉलपेपर
- 5.READ_VOICEMAIL

Runtime / Dangerous Permissions

खतरनाक अनुमतियों में वे अनुमतियाँ शामिल हैं जो किसी तरह से उपयोगकर्ता के डेटा को प्रभावित करती हैं। उदाहरण के लिए, यदि आप फ़ोन से संपर्क पढ़ना चाहते हैं या फ़ोन के फ़ाइल संग्रहण तक पहुँचना चाहते हैं, तो ये अधिकार खतरनाक श्रेणी में आते हैं क्योंकि इनमें उपयोगकर्ता की गोपनीयता शामिल होती है। खतरनाक अनुमतियों का उपयोग करने के लिए, आपको पहले अलर्ट डायलॉग या कोई अन्य डायलॉग प्रदर्शित करके स्पष्ट रूप से अनुमति लेनी होगी। यदि उपयोगकर्ता अनुमति से इनकार करता है, तो आपका एप्लिकेशन उस अनुमति का उपयोग करने में असमर्थ होगा।

1.पढ़े_कैलेंडर

2.कैलेंडर लिखें

3.कैमरा

4.कॉल_लॉग_पढ़ें

5.WRITE_CALL_LOG

6.पढ़े_संपर्क

7.संपर्क लिखें

8.GET_खाते

9.ACCESS_FINE_LOCATION

10.पहुंच_मोटे_स्थान

11.एसएमएस भेजें

Android UI Controls

T	TextView	
T	EditText	
Button	Button	
	ImageButton	
	ToggleButton	
	RadioButton	
	RadioGroup	
	RatingBar	
	CheckBox	
	ProgressBar	
	Spinner	
	TimePicker	
	DatePicker	
	SeekBar	
	AlertDialog	
	Switch	



S.No.	UI control और description
1.	Textview : इस नियन्त्रण का उपयोग user को पाठ को प्रदर्शित करने के लिए किया जाता है
2.	AutoCompleteTextView : AutoCompleteTextView एक प्रकार का दृश्य होता है जो की editText के समान होता है इसके आलावा यह user को type को करते समय पूरा होने वाले सुझावों की एक सूचि दिखता है
3.	Button : एक पुश button होता है जिसे user के द्वारा कारवाई करने के लिये दबाया या क्लिक किया जा सकता है
4.	EditText : edittext ,textview का एक पूर्व निर्धारित sub-class होता है जिसमें समृद्ध क्षमताये शामिल होती है
5.	ImageButton : एक imagebutton एक absolutelayout होता है जो आपको स्टीक स्थान को निर्दिष्ट करने में सक्षम बनाता है यह छवि के साथ एक button को दिखाता है (पाठ के बजाय) जिसे user के द्वारा दबाया या क्लिक किया जा सकता है

S.No.	UI control और description
6.	CheckBox : एक on/ off switch होता है जिसे user के द्वारा on किया जा सकता है users को चयन योग्य आप्शनस के समूह के साथ प्रस्तुत करते समय आपको check box का उपयोग करना चाहिए जो पारस्परिक रूप से अनन्य (mutually exclusive) नहीं है
7.	ToggleButton : प्रकाश संकेतक के साथ एक on/off button
8.	RadioButton : RadioButton की दो अवस्थाएं होते हैं या तो checked या unchecked
9.	RadioGroup : RadioGroup का उपयोग एक या अधिक RadioButton को एक साथ समूहीकृत करने के लिए किया जाता है

S.No.	UI control और description
10.	ProgressBar : ProgressBar के sight(दृश्य) कुछ चल रहे कार्यों के बारे में sight(दृश्य) प्रतिक्रिया प्रदान करता है जैसा कि जब आप background में कोई भी कार्य कर रहे होते हैं
11.	Spinner : एक drop-down सचि जो कि users को एक सेट से एक मूल्य का चयन करने की अनुमति देती है
12.	TimePicker : TimePicker के sight(दृश्य) users को 24-hours mode या AM/PM mode में दिन के समय का चयन करने में सक्षम बनाता है
13.	DatePicker : DatePicker के sight(दृश्य) users को दिन की एक तारीख का चयन करने में सक्षम बनाता है

Event Handling

इवेंट, एप्लीकेशन के इंटरैक्टिव घटकों के साथ उपयोगकर्ता की बातचीत के बारे में डेटा एकत्र करने का एक उपयोगी तरीका है। जैसे बटन प्रेस या स्क्रीन टच आदि। एंड्रॉइड फ्रेमवर्क फर्स्ट-इन, फर्स्ट-आउट (FIFO) आधार पर इवेंट कतार बनाए रखता है। आप इन इवेंट को अपने प्रोग्राम में कैप्चर कर सकते हैं और ज़रूरत के हिसाब से उचित कार्रवाई कर सकते हैं।

There are following three concepts related to Android Event Management

- **Event Listeners** – यह व्यू क्लास में एक इंटरफ़ेस है। इसमें एक एकल कॉलबैक विधि होती है। जब श्रोता जिस दृश्य से जुड़ा होता है, वह उपयोगकर्ता इंटरैक्शन के कारण ट्रिगर हो जाता है, तो कॉलबैक विधियाँ कॉल की जाती हैं।
- **Event Listeners Registration** – इवेंट पंजीकरण वह प्रक्रिया है जिसमें एक इवेंट हैंडलर एक इवेंट श्रोता के साथ संबद्ध हो जाता है ताकि जब संबंधित इवेंट श्रोता इवेंट को फायर करता है तो इस हैंडलर को कॉल किया जाता है।
- **Event Handlers** – यह उस इवेंट से निपटने के लिए जिम्मेदार होता है जिसके लिए इवेंट श्रोताओं ने पंजीकरण किया है और उस संबंधित इवेंट के लिए वांछित कार्रवाई करता है।

Event Listeners & Event Handlers

Event Handler	Event Listener & Description
onClick()	OnClickListener() इसे तब call किया जाता है जब उपयोगकर्ता किसी विजेट जैसे बटन, टेक्स्ट, छवि आदि पर क्लिक करता है, स्पर्श करता है या फोकस करता है। आप ऐसे इवेंट को संभालने के लिए onClick() इवेंट हैंडलर का उपयोग करेंगे।
onLongClick()	OnLongClickListener() इसे तब कॉल किया जाता है जब उपयोगकर्ता किसी विजेट जैसे बटन, टेक्स्ट, इमेज आदि पर एक या अधिक सेकंड के लिए क्लिक करता है या छूता है या फ़ोकस करता है। आप ऐसे इवेंट को हैंडल करने के लिए onLongClick() इवेंट हैंडलर का उपयोग करेंगे।
onFocusChange()	OnFocusChangeListener() इसे तब कॉल किया जाता है जब विजेट अपना फोकस खो देता है यानी उपयोगकर्ता व्यू आइटम से दूर चला जाता है। आप इस तरह की घटना को संभालने के लिए onFocusChange() इवेंट हैंडलर का उपयोग करेंगे।
onKey()	OnKeyListener() इसे तब कॉल किया जाता है जब उपयोगकर्ता आइटम पर ध्यान केंद्रित करता है और डिवाइस पर हार्डवेयर कंजी दबाता है या छोड़ता है। आप इस तरह की घटना को संभालने के लिए onKey() इवेंट हैंडलर का उपयोग करेंगे।

Event Listeners & Event Handlers

Event Handler	Event Listener & Description
onTouch()	OnTouchListener() इसे तब बुलाया जाता है जब उपयोगकर्ता कुंजी दबाता है, कुंजी छोड़ता है, या स्क्रीन पर कोई भी हरकत करता है। आप ऐसे इवेंट को संभालने के लिए onTouch() इवेंट हैंडलर का उपयोग करेंगे।
onMenuItemClick()	OnMenuItemClickListener() इसे तब कॉल किया जाता है जब उपयोगकर्ता कोई मेनू आइटम चुनता है। आप ऐसे इवेंट को हैंडल करने के लिए onMenuItemClick() इवेंट हैंडलर का उपयोग करेंगे।
onCreateContextMenu()	onCreateContextMenuItemListener() इसे तब कहा जाता है जब संदर्भ मेनू बनाया जा रहा हो (लगातार "लंबे क्लिक" के परिणामस्वरूप)

File Edit View Navigate Code Refactor Build Run Tools Git Window Help Event Handler - MainActivity.java [Event_Handler.app.main]

EventHandler > app > src > main > java > com > example > eventhandler > MainActivity

MainActivity.java activity_main.xml

Device Manager Virtual Physical Create device Device Emulator: Pixel 4a API 30 API Size on Disk Actions

Resource Manager Pull Requests

Bookmarks Build variants

Structure

Git Run TODO Profiler Problems Terminal Logcat App Inspection

00:00 00:16

```
package com.example.eventhandler;

import androidx.appcompat.app.AppCompatActivity;
import android.graphics.Color;
import android.os.Bundle;
import android.view.View;
import android.widget.ImageButton;
import android.widget.TextView;

public class MainActivity extends AppCompatActivity {
    ImageButton ib1, ib2;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        ib1=(ImageButton) findViewById(R.id.imageButton6);
        ib2=(ImageButton) findViewById(R.id.imageButton7);

        ib1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                TextView tv = (TextView) findViewById(R.id.textView5);
                tv.setTextColor(getResources().getColor(R.color.red));
            }
        });
        ib2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                TextView tv = (TextView) findViewById(R.id.textView5);
                tv.setTextColor(getResources().getColor(R.color.green));
            }
        });
    }
}
```

Android UI Layouts

एंड्रॉइड लेआउट का उपयोग यूजर इंटरफ़ेस को define करने के लिए किया जाता है जो UI controls या widgets रखता है जो एंड्रॉइड एप्लिकेशन या activity screen की स्क्रीन पर दिखाई देगा। आम तौर पर, हर एप्लिकेशन व्यू और व्यूग्रुप का combination होता है। जैसा कि हम जानते हैं, एक एंड्रॉइड एप्लिकेशन में बड़ी संख्या में गतिविधियाँ होती हैं और हम कह सकते हैं कि प्रत्येक गतिविधि एप्लिकेशन का एक पेज है। इसलिए, प्रत्येक गतिविधि में कई यूजर इंटरफ़ेस components होते हैं और वे components व्यू और व्यूग्रुप के उदाहरण होते हैं। लेआउट में सभी तत्व व्यू और व्यूग्रुप ऑब्जेक्ट के hierarchy का उपयोग करके बनाए जाते हैं।

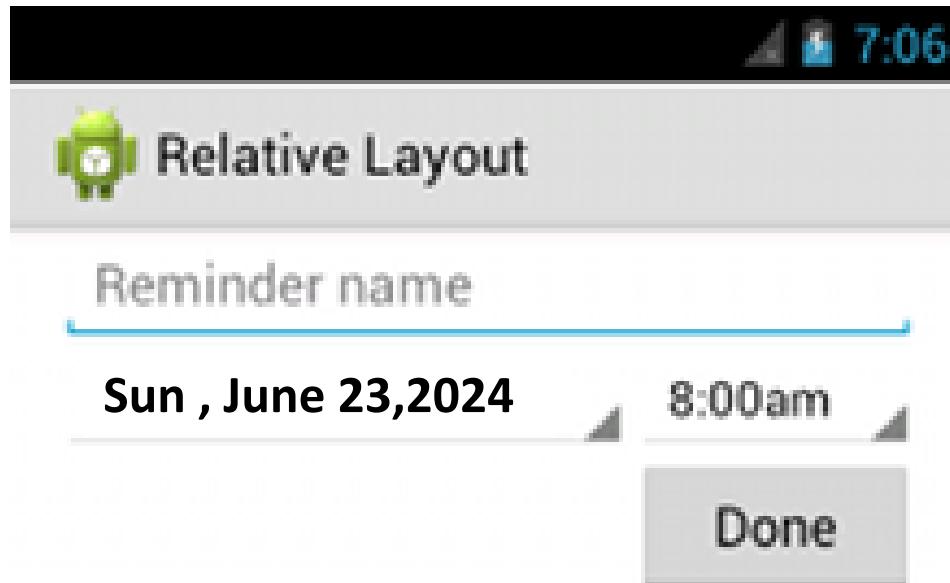
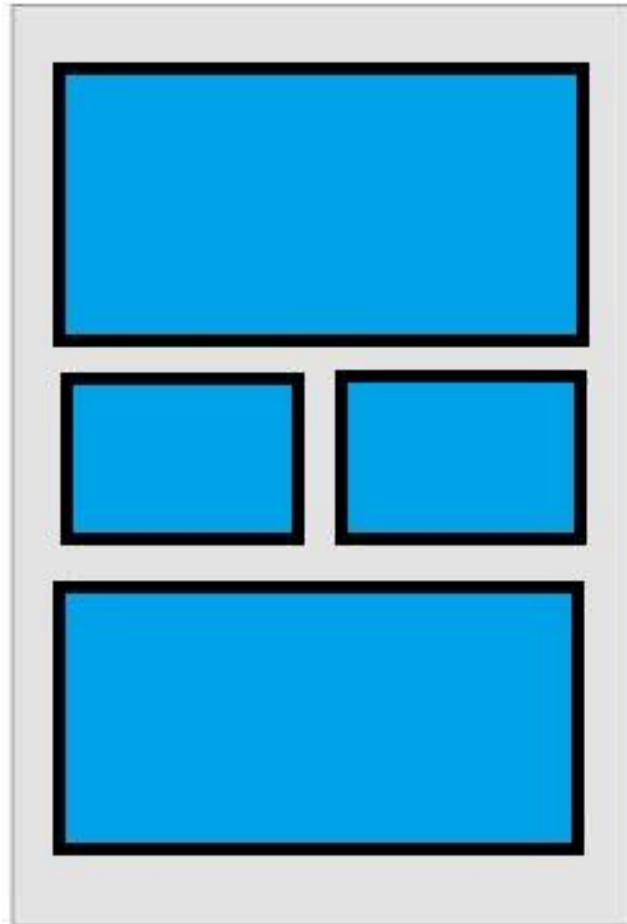
Types of Android Layout

- Android Linear Layout
- Android Relative Layout
- Android Constraint Layout
- Android Frame Layout
- Android Table Layout
- Android Web View
- Android List View
- Android Grid View

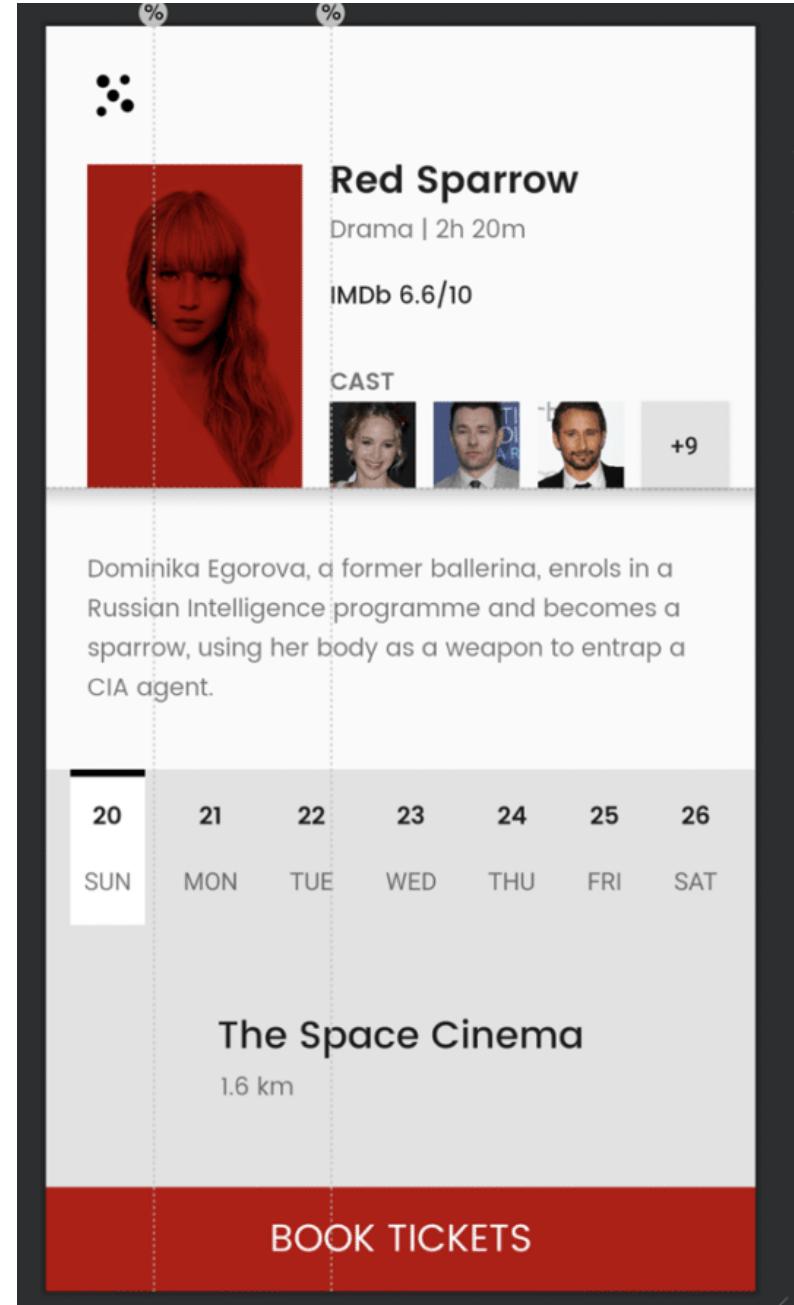
•एंड्रॉइड लीनियर लेआउट: लीनियर लेआउट एक व्यूग्रुप उपवर्ग है, जिसका उपयोग ओरिएंटेशन प्रॉपर्टी के आधार पर चाइल्ड व्यू तत्वों को एक-एक करके किसी विशेष दिशा में क्षैतिज या लंबवत रूप से प्रदान करने के लिए किया जाता है।



• एंड्रॉइड रिलेटिव लेआउट: RelativeLayout एक ViewGroup उपवर्ग है, जिसका उपयोग चाइल्ड व्यू तत्वों की एक दूसरे के सापेक्ष स्थिति को निर्दिष्ट करने के लिए किया जाता है (जैसे A को B के दाईं ओर) या पैरेंट के सापेक्ष (पैरेंट के शीर्ष पर स्थिर)।



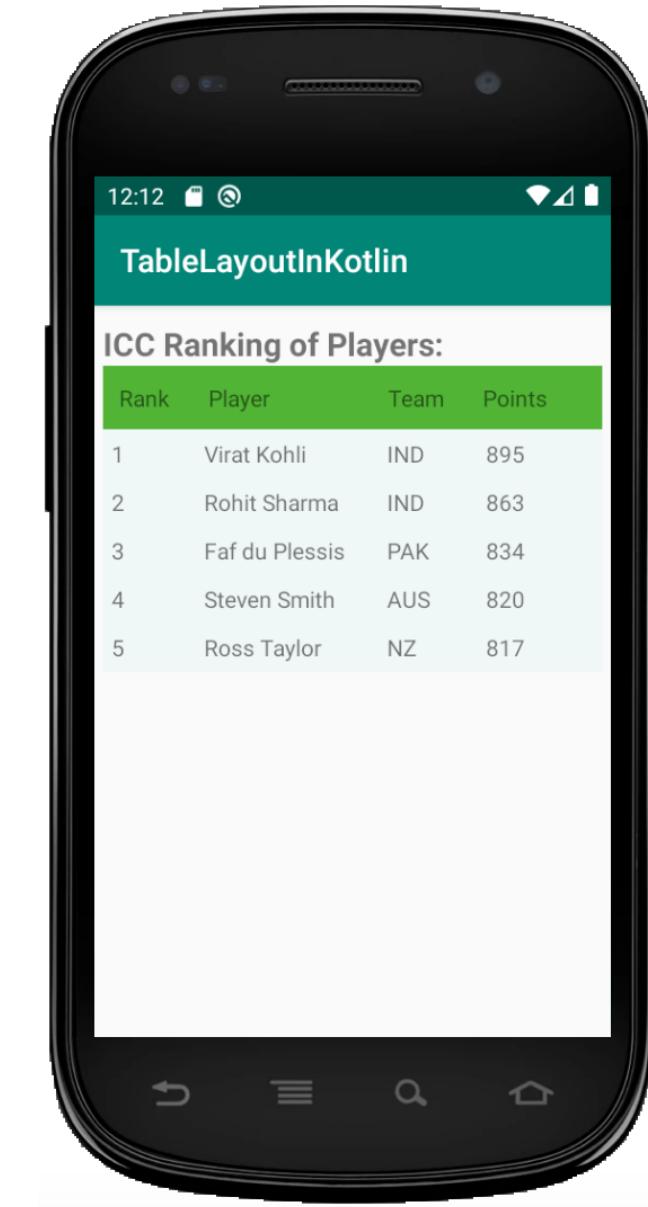
•**एंड्रॉड कंस्ट्रेन्ट लेआउट:** कंस्ट्रेन्ट लेआउट एक व्यूग्रुप सबकलास है, जिसका उपयोग हर चाइल्ड व्यू के लिए लेआउट कंस्ट्रेन्ट की स्थिति को अन्य मौजूद व्यू के सापेक्ष निर्दिष्ट करने के लिए किया जाता है। एक कंस्ट्रेन्ट लेआउट एक रिलेटिव लेआउट के समान है, लेकिन इसमें अधिक शक्ति होती है।



•एंड्रॉइड फ्रेम लेआउट: फ्रेमलेआउट एक व्यूग्रुप उपवर्ग है, जिसका उपयोग फ्रेमलेआउट के अंदर केवल एक ही दृश्य प्रदर्शित करने के लिए इसमें शामिल व्यू तत्वों की एक दूसरे के शीर्ष पर स्थिति निर्दिष्ट करने के लिए किया जाता है।



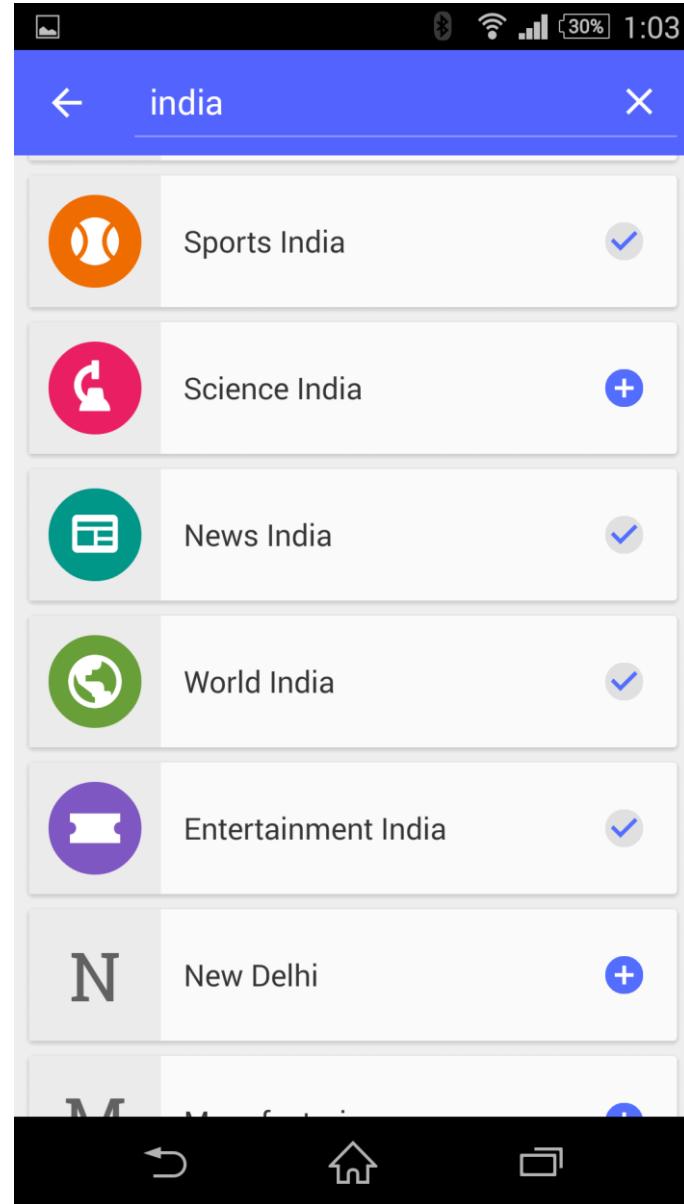
•**एंड्रॉइड टेबल लेआउट:** टेबललेआउट एक व्यूग्रुप उपवर्ग है, जिसका उपयोग पंक्तियों और स्तंभों में चाइल्ड व्यू तत्वों को प्रदर्शित करने के लिए किया जाता है।



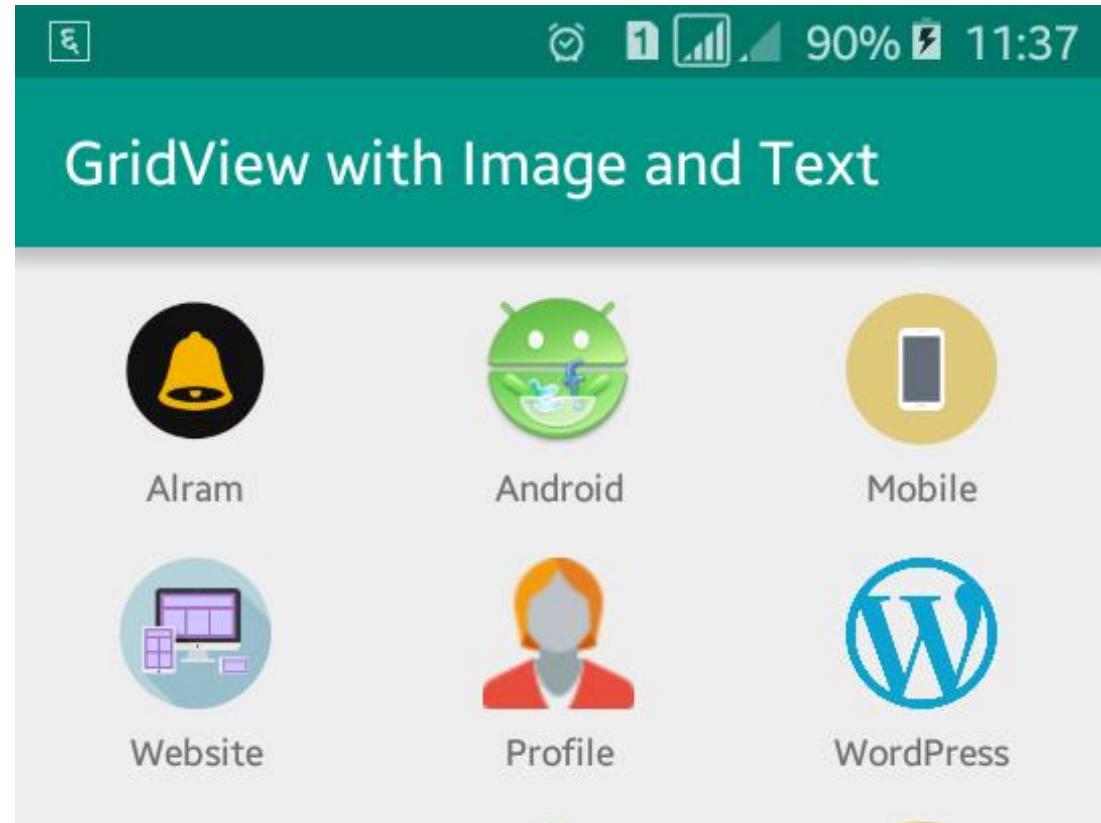
•**एंड्रॉइड वेब व्यूः** वेबव्यू एक ब्राउज़र है जिसका उपयोग हमारे गतिविधि लेआउट में वेब पेजों को प्रदर्शित करने के लिए किया जाता है।



•**एंड्रॉयड सूची दृश्य:** सूची दृश्य एक दृश्य समूह है, जिसका उपयोग एकल कॉलम में आइटमों की स्क्रॉल करने योग्य सूची प्रदर्शित करने के लिए किया जाता है।



•**एंड्रॉइड ग्रिड व्यूः** ग्रिड व्यू एक व्यूग्रुप है जिसका उपयोग पंक्तियों और स्तंभों के ग्रिड दृश्य में आइटमों की स्क्रॉल करने योग्य सूची प्रदर्शित करने के लिए किया जाता है।



Intents

Intents का मतलब होता है आप क्या करना चाहते हैं।

जैसे कि

- मैं contacts list देखना चाहता हूँ।
- मैं कैमरा से फोटो किलक करना चाहता हूँ।
- मैं messages शो करना चाहता हूँ।

आप क्या करना चाहते हैं ये आप इंटेंट क्रिएट करके एक मैसेज के द्वारा बताते हैं।

Intents messages होते हैं जो android components(Activity, Service, Broadcast receiver, Content provider) एक दूसरे को भेजते हैं। एक component किसी दूसरे component को किसी functionality के लिए request करता है। ये request messages के द्वारा की जाती है जिन्हे intents कहते हैं। Intents asynchronous messages होते हैं क्योंकि android components बिना wait किये एक दूसरे को ये messages send करते हैं।

Types of Intents

Intents 2 तरह के होते हैं

- Implicit intent
- Explicit intent

Implicit Intent

जैसे कि आपने एक intent क्रिएट किया जो कि एक web address को ओपन करता है। लेकिन android system में एक से ज्यादा browsers हो सकते हैं। यदि सिर्फ एक target (Browser) है तो directly वही open हो जायेगा नहीं तो जितने browsers हैं उनकी लिस्ट show हो जाएगी जिसमे से यूज़र सेलेक्ट कर सकता है और उसी टारगेट के साथ intent execute हो जायेगा।

Explicit Intent

Explicit intents में आप पहले ही डिफाइन कर देते हैं कि intent किस टारगेट के साथ execute होना है। जैसे कि आपने एक intent क्रिएट किया जो एक वेब एड्रेस ओपन करता है तो आप इसमें पहले ही डिफाइन कर देते हैं कि ये इंटेंट Google Chrome के साथ execute होना है। इसमें हम टारगेट कॉम्पोनेन्ट पहले से ही डिफाइन कर देते हैं।

Android Menus

Menus किसी भी एप्लीकेशन का एक बहुत ही कॉमन इंटरफ़ेस होता है। यूजर को easy और designer यूजर इंटरफ़ेस प्रोवाइड करने के लिए आप menus को यूज कर सकते हैं। Menus को आप नीचे दिए गए 3 reasons की वजह से use कर सकते हैं।

- User को application के दूसरे components की link provide करने के लिए।
- Application को structure प्रोवाइड करने के लिए।
- easy navigation के लिए।

Android में menus 3 तरह की होते हैं।

1.Options menus

2.Context menus

3.Popup menus

Creating a Menu in XML Layout File

सभी menu types के लिए menu items डिफाइन करने के लिए android एक स्टैण्डर्ड XML format प्रोवाइड करता है। अपनी activity के कोड में menu क्रिएट करने की बजाए यदि आप चाहे तो आप एक XML layout file में भी menus बना सकते हैं। बाद में आप इसे अपनी activity में load कर सकते हैं। ऐसी XML layout file को menu resource कहते हैं।

इसके लिए सबसे पहले आपको अपने प्रोजेक्ट की res/menu directory में एक XML फाइल क्रिएट करनी होगी।

XML file में menu क्रिएट करने के लिए आप कुछ elements को यूज़ करते हैं

<menu> - ये एलिमेंट menu define करता है। ये element menu items के base का काम करता है। आपकी file में सबसे पहले ये एलिमेंट ही declare होना चाहिए। इसके अंदर आप **<item>** और **<group>** एलिमेंट डाल सकते हैं।

<item> - ये एलिमेंट एक menu item क्रिएट करता है। हर item के लिए आपको एक item element declare करना होगा। <item> element कुछ attributes यूज़ करता है जो की नीचे दिए जा रहे हैं।

•**android : id** - ये एक यूनिक id होती है जिसे android system menu item को uniquely identify करता है।

•**android : icon** - इस attribute के द्वारा आप किसी image को item के icon की तरह use कर सकते हैं।

•**android : title** - इस attribute में आप item का title declare करते हैं।

<group> - ये element भी container की तरह ही होता है, लेकिन ये एक जैसे menu items को एक साथ रखने के लिए use किया जाता है।

```
<menu xmlns:android="schema address">  
    <item android:id="1"  
        android:icon="address of icon image"  
        android:title="First Item">  
        <item android:id="2"  
            android:icon="address of icon image"  
            android:title="Second Item">  
            <item android:id="3"  
                android:icon="address of icon image"  
                android:title="Third Item">  
    </menu>
```

Options Menus

किसी भी application के लिए options menus primary menus होती है। Options menu में आपको वो options डालने चाहिए जो पूरी application को control करते हैं। जैसे कि सर्च और सेटिंग्स का option आप options menus में डाल सकते हैं। यदि आप android gingerbread या उससे lower versions के लिए एप्लीकेशन develop कर रहे हैं तो यूज़र menu बटन पर क्लिक करके menus को देख सकता है।

यदि आप android honeycomb या उससे ऊपर के versions के लिए options menus develop कर रहे हैं तो यूज़र app bar के द्वारा menus को access कर सकते हैं।

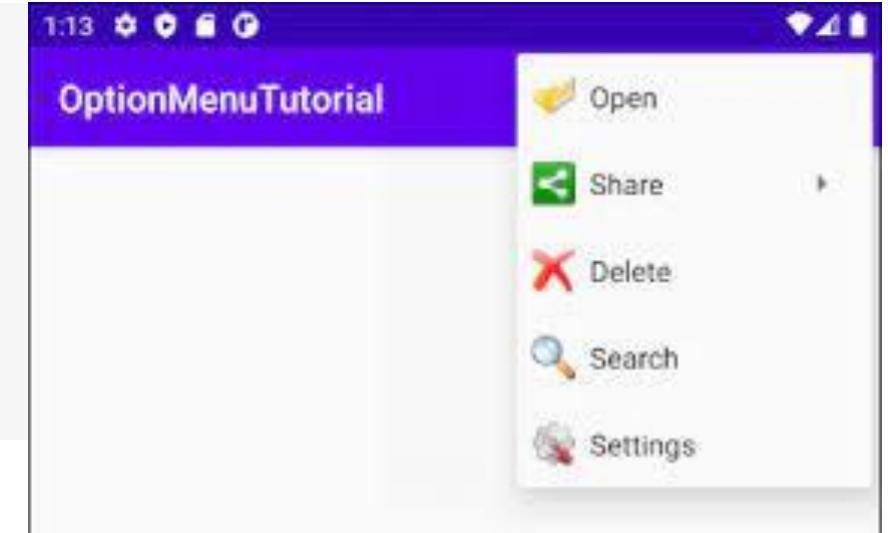
Creating Options Menus

Options menu के लिए items आप या तो activity subclass से या fragment subclass से डिक्लेअर कर सकते हैं। यदि आप दोनों तरीके से items डिक्लेअर करते हैं तो यूज़र interface में दोनों combine हो जाते हैं।

Activity के item पहले show होते हैं इसके बाद जिस order में fragments को ऐड किया गया था उसी आर्डर में उनके items शो होते हैं। यदि आप चाहे तो menu items को reorder भी कर सकते हैं।

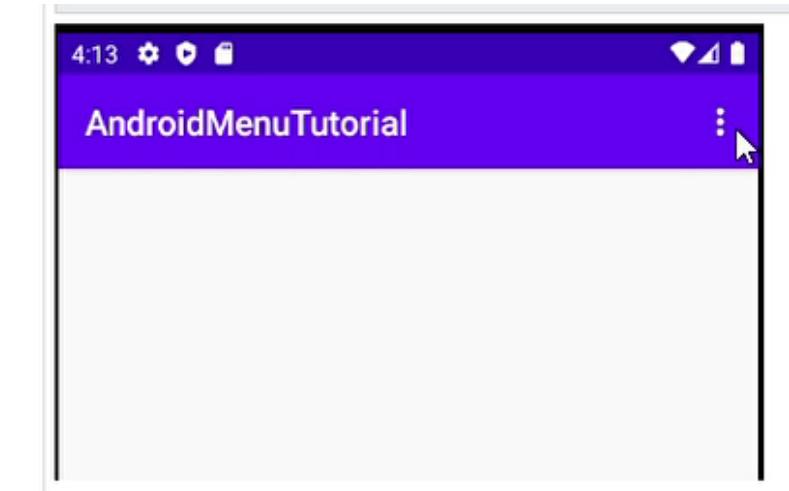
किसी भी activity में options menu define करने के लिए आपको onCreateOptionsMenu() मेथड को override करना होता है।

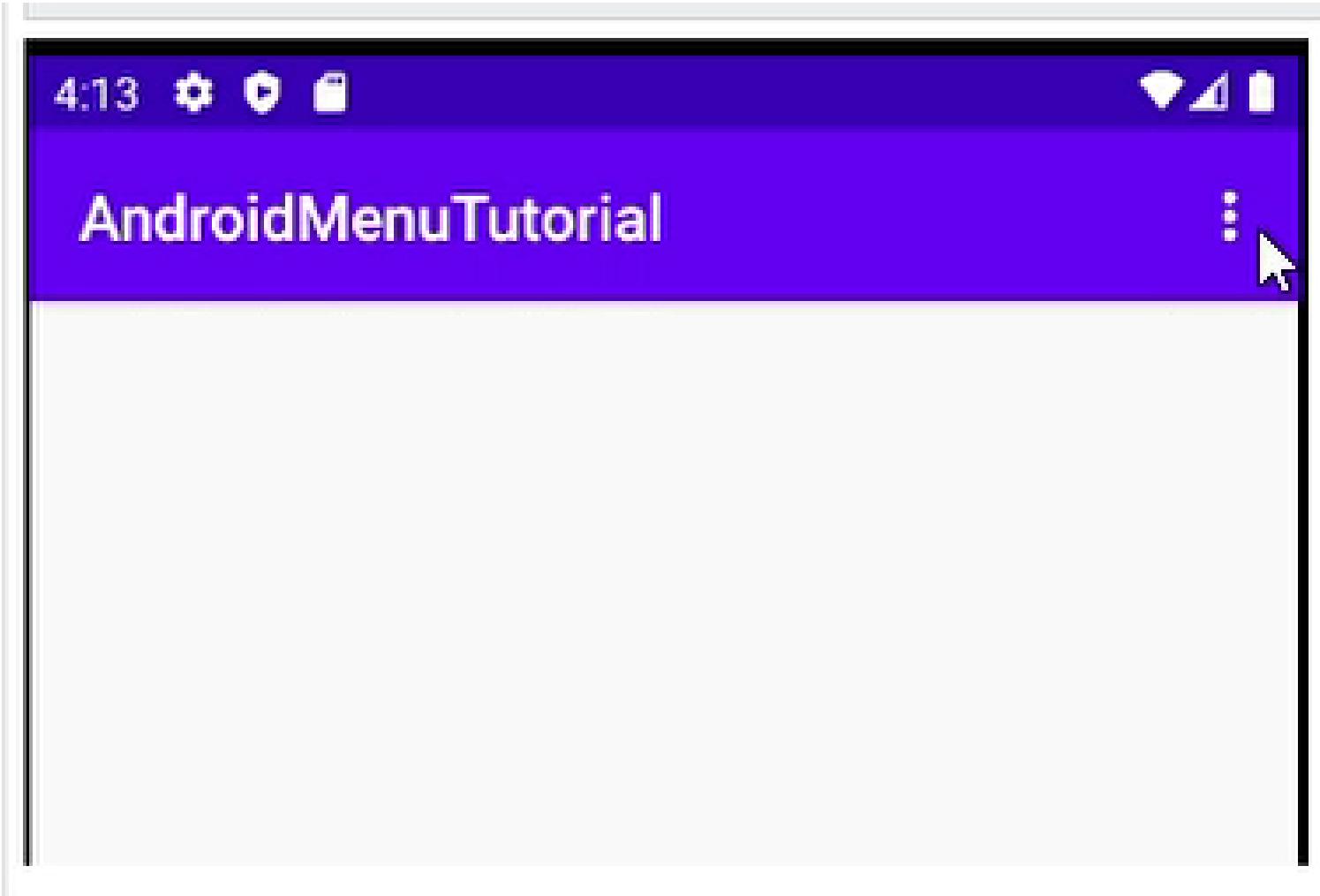
```
@Override  
public boolean onCreateOptionsMenu(Menu menu)  
{  
// put menu resources in menu from XML  
}
```



इस मेथड के द्वारा आप मेनू resources को भी मेनू में डाल सकते हैं। आप चाहे तो आप add() मेथड के द्वारा items को add भी कर सकते हैं। और मेनू में items को ढूँढने के लिए आप findItem() मेथड यूज़ कर सकते हैं।

```
<?xml version="1.0" encoding="utf-8"?>
<menu
    xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:title="Menu Item 1" />
    <item android:title="Menu Item 2" >
        <menu >
            <item android:title="Menu Item 2.1" />
            <item android:title="Menu Item 2.2" />
        </menu>
    </item>
    <item android:title="Menu Item 3" />
</menu>
```



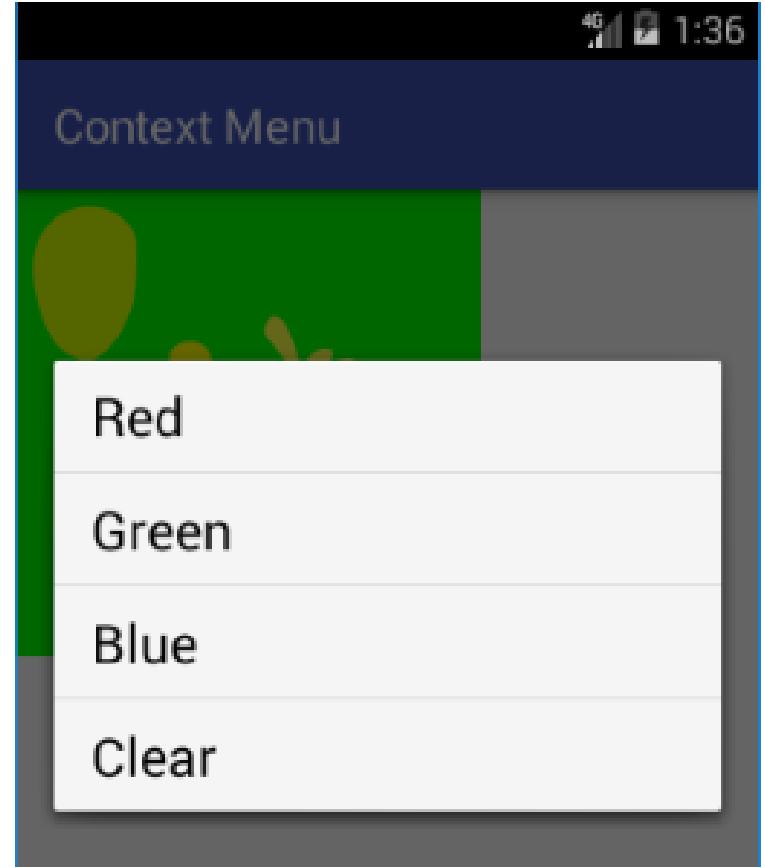


Context Menus

एक context menu वो option शो करती है जो किसी एक एलिमेंट को effect करती है। ये एलिमेंट वो होता है जिसे यूज़र ने long click करके छोड़ा है।

Context menus movable menus होती है। जब यूज़र बहुत देर तक किसी एलिमेंट पर टच करके छोड़ता है तो ये menu शो होती है। ये उस element को effect करने वाले ऑप्शन प्रोवाइड करती है जिस पर यूज़र ने long क्लिक किया है।

Context menu क्रिएट करने के लिए सबसे पहले आपको जिस view के लिए आप context menu क्रिएट करना चाहते हैं उसको register करना होगा। इसके लिए आपको registerForContextMenu() मेथड कॉल करना होगा और उसमे view पास करना होगा। इसके बाद आपको onCreateContextMenu() मेथड को ऑवरराइड करना होगा। जब भी यूज़र registered view पर long क्लिक करता है तो system इस मेथड को कॉल करता है। इसी मेथड में आप menu items define करते हैं।



```
@Override  
public void onCreateContextMenu(ContextMenu contextmenu, View view, ContextMenuInfo  
contextmenuinfo)  
{  
// declare menu items and put menu resources  
}
```

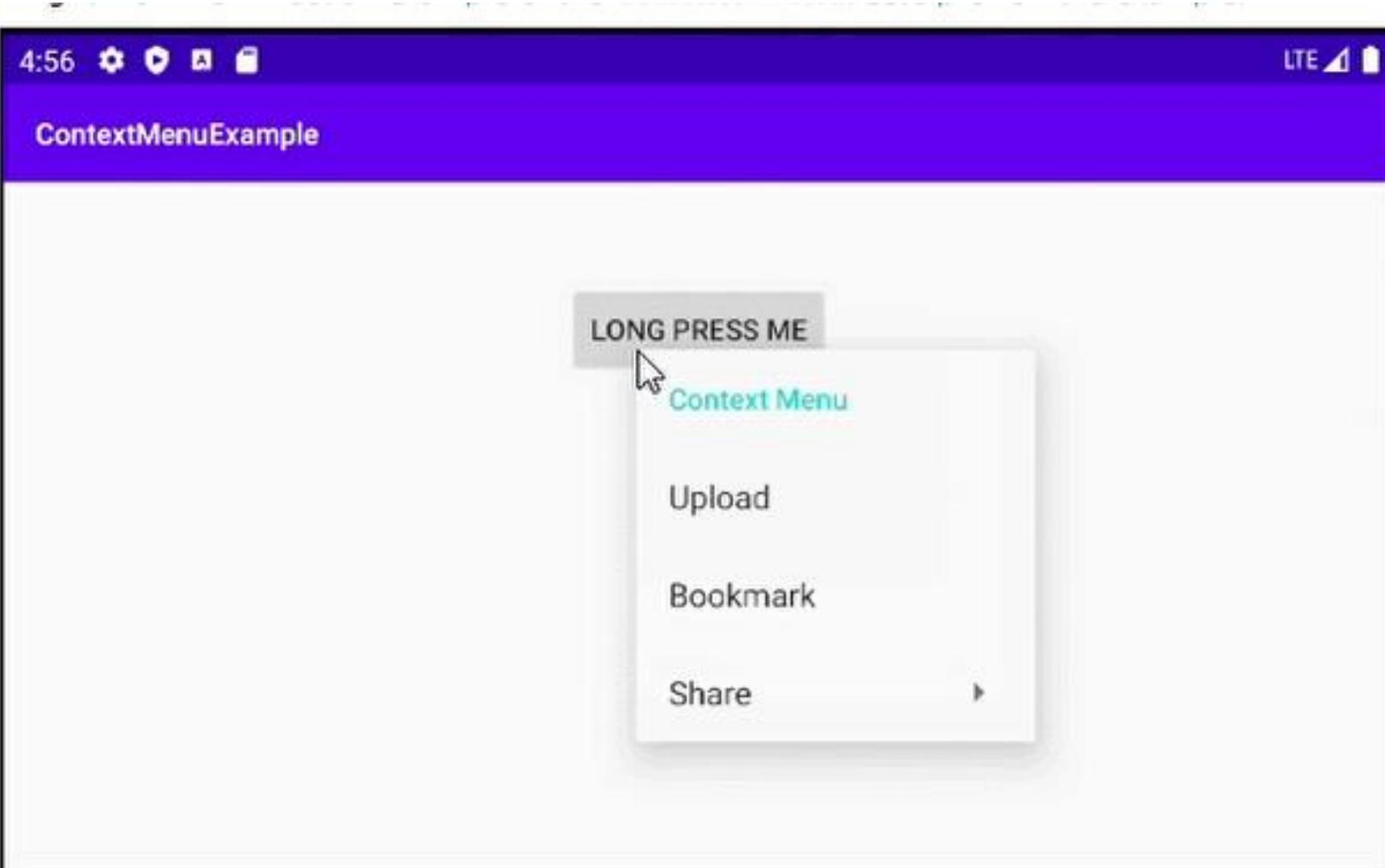
इसके बाद आपको onContextItemSelected() मेथड को ओवरराइड करना होगा। जब भी कोई यूज़र menu में से कोई item select करता है system इस मेथड को कॉल करता है। आप item select होने पर जो task complete करना चाहते हैं वो इस मेथड में कर सकते हैं।

```
@Override  
public boolean onContextItemSelected(MenuItem item)  
{  
// perform desired task  
}
```

```
@Override  
protected void onCreate(Bundle savedInstanceState)  
{  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
  
    this.button = (Button) this.findViewById(R.id.button_test);  
  
    this.registerForContextMenu(this.button);  
}
```

```
@Override  
public void onCreateContextMenu(ContextMenu menu, View view,  
ContextMenu.ContextMenuInfo menuInfo)  
  
{  
    super.onCreateContextMenu(menu, view, menuInfo);  
  
    menu.setHeaderTitle("Context Menu");  
  
    MenuItemInflater inflater = getMenuInflater();  
    inflater.inflate(R.menu.layout_context_menu, menu);  
}
```

```
// You may not need "Android Resource File" to have a ContextMenu.  
// Using Java to create Context Menu.  
public void onCreateContextMenu_2(ContextMenu menu, View view,  
ContextMenu.ContextMenuItemInfo menuInfo)  
  
{  
    super.onCreateContextMenu(menu, view, menuInfo);  
  
    menu.setHeaderTitle("Context Menu");  
  
    // groupId, itemId, order, title  
    MenuItem menuItemUpload = menu.add(1, 1, 1, "Upload");  
    MenuItem menuItemBookmark = menu.add(2, 2, 2, "Bookmark");  
  
    // groupId, itemId, order, title  
    SubMenu subMenuShare= menu.addSubMenu(3, 3, 3, "Share");  
    subMenuShare.add(4, 31, 1, "Google" );  
    subMenuShare.add(5, 32, 2, "Instagram");  
}
```



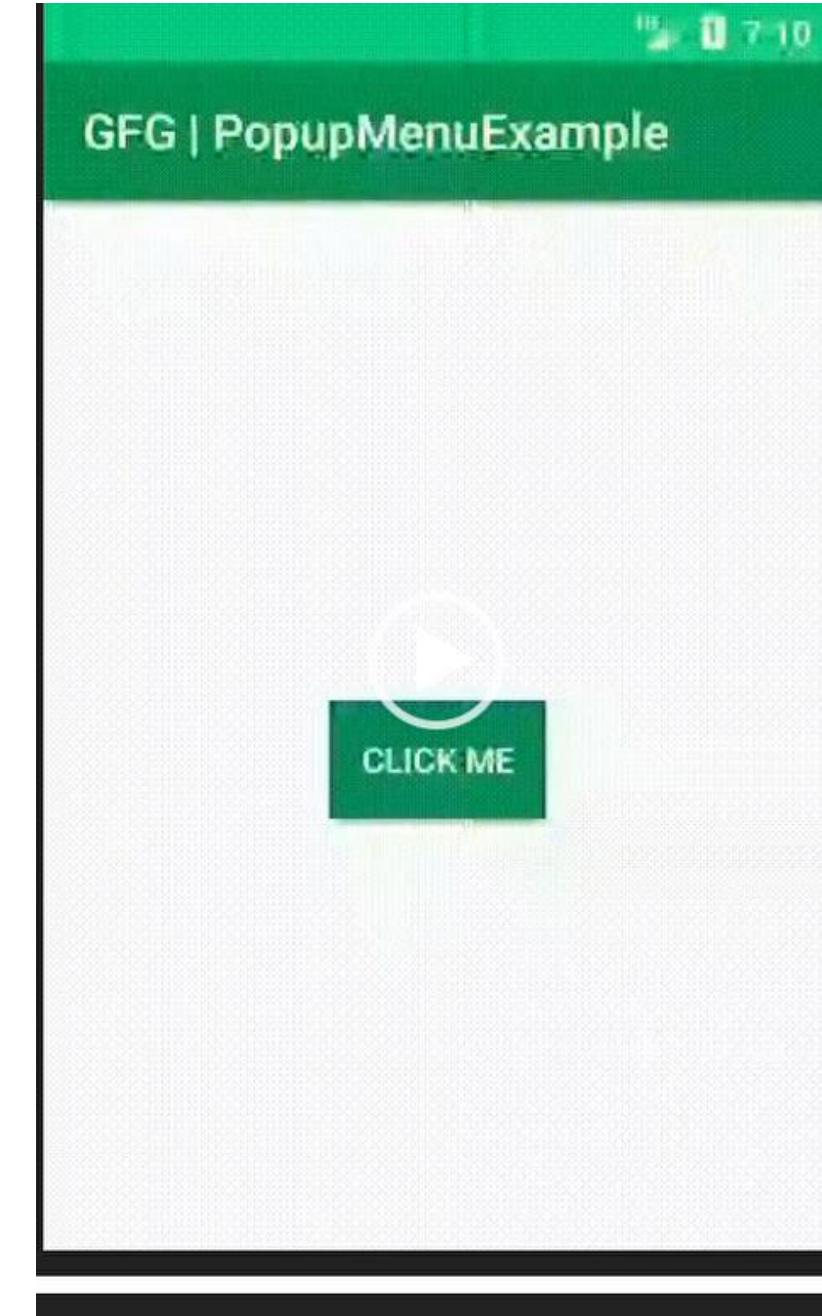
Popup Menus

एक popup menu , items को vertical list में शो करती है। Popup menus anchor view के साथ शो होती है। अगर जगह available होती है तो ये anchor view के नीचे show होती है नहीं तो anchor view के उपर शो होती है।

Popup menus ऐसे options प्रोवाइड करती है जो same activity में दूसरे views के साथ perform किये जा सकते हैं।

Creating Popup Menu

सबसे पहले आपको PopupMenu क्लास का एक object क्रिएट करना होगा। Object क्रिएट करते समय आप उसमे current application का context और वो view pass करेंगे जिसमे आप popup menu लगाना चाहते हैं। इसके बाद PopupMenu.getMenu() मेथड कॉल करके आप को Menu का reference लेना होगा और MenuInflater की द्वारा menu में resources put करने होंगे। सबसे आखिर में menu को शो करने के लिए PopupMenu.show() method कॉल करना होगा।



ListView in Android

एंड्रॉयड में लिस्ट व्यू एक प्रकार का व्यूग्रुप है जो आइटम की सूची को पंक्तियों के रूप में दिखाता है। लिस्ट व्यू में लिस्ट लेआउट लिस्ट व्यू का उपयोग करके बनाया जाता है। यह एंड्रॉयड में सबसे बुनियादी लेकिन सबसे महत्वपूर्ण यूआई घटकों में से एक है। यह सूची को लंबवत रूप में दिखाता है और यह स्वचालित रूप से स्क्रॉल करने योग्य हो जाता है।

आइटम Adapter का उपयोग करके स्वचालित रूप से जोड़े जाते हैं। Adapter मूल रूप से आइटम के लिए UI घटक और डेटा स्रोत के बीच पुल का काम करता है। Adapter डेटा को होल्ड करता है और उसे Adapter व्यू में भेजता है, उसके बाद व्यू Adapter व्यू से डेटा लेता है और उसे ListViews में दिखाता है।



Parameters of ListView

हम **onListItemClick()** विधि का उपयोग करके सूची आइटम पर क्लिक किए जाने के बाद processed की जाने वाली action भी सेट कर सकते हैं। इस विधि में, 4 पैरामीटर पास किए जाते हैं:

- 1.ListView:** इसमें view का आइटम शामिल होता है।
- 2.View:** यह selected specific view बताता है।
- 3.Position:** यह सारणी में चयनित आइटम की स्थिति बताता है।
- 4.Id:** Id is used to uniquely identify a view.

- To add a ListView in our application we define it using the following:

1. <ListView xmlns:android= “http://schemas.android.com/apk/res/android”
2. xmlns:tools= “http://schemas.android.com/tools”
3. android:id= “@+id/MyLV”
4. >

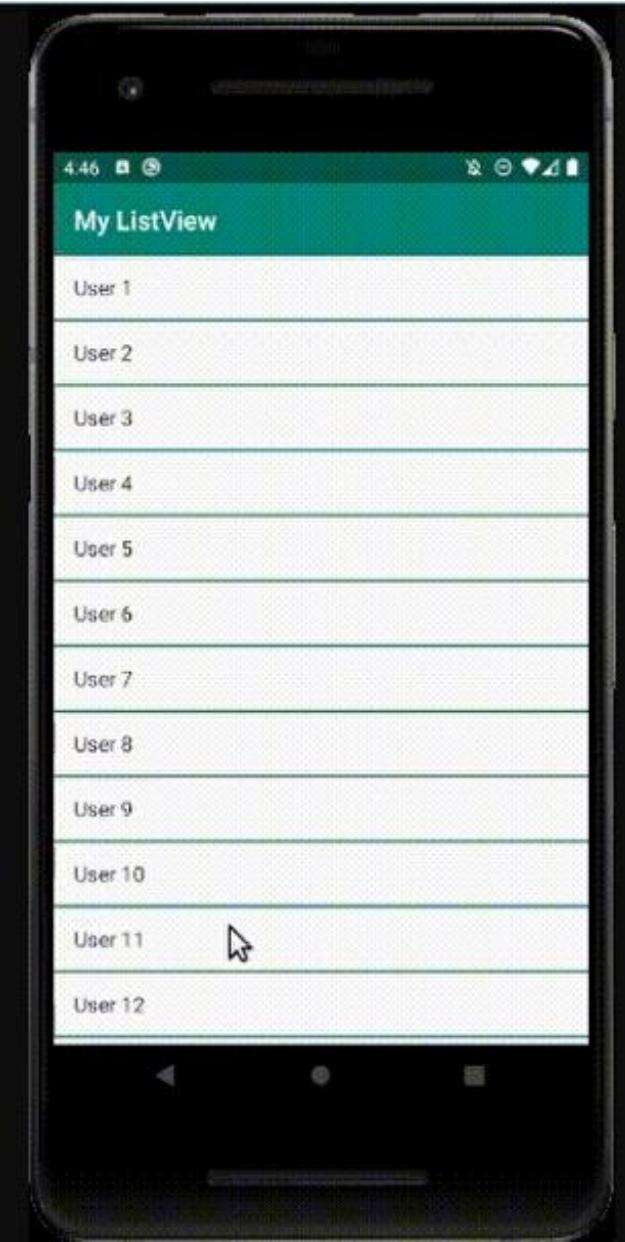
एंड्रॉयड में ListView की विशेषताएँ

एंड्रॉयड में ListView को अनुकूलित करने के लिए, हम इसमें कुछ विशेषताएं जोड़ सकते हैं, जैसे:

- 1.आईडी: आईडी विशिष्ट रूप से सूची दृश्य की पहचान करती है।
- 2.विभाजक: यह विभिन्न सूची आइटमों के बीच रंग या ड्राएबल को परिभाषित करता है।
- 3.डिवाइडर ऊंचाई: यह सूची के बीच विभाजक की ऊंचाई को परिभाषित करता है।
- 4.Items.listSelector: यह गुण listView के लिए चयनकर्ता सेट करता है।
- 5.प्रविष्टियाँ: यह सरणी संसाधन का संदर्भ निर्दिष्ट करता है।

एंड्रॉयड में ListView की विशेषताएँ

6. **पृष्ठभूमि:** यह ListView की पृष्ठभूमि सेट करता है। पृष्ठभूमि रंग या छवि भी हो सकती है।
7. **चॉइसमोड:** यह एक बार में चयनित की जा सकने वाली वस्तुओं की संख्या निर्धारित करता है।
8. **TextSize:** यह प्रदर्शित होने वाले पाठ का आकार निर्धारित करता है।
9. **लेआउट_चौड़ाई:** यह लेआउट की चौड़ाई निर्धारित करता है।
10. **लेआउट_ऊंचाई:** यह लेआउट की ऊंचाई निर्धारित करता है।
11. **लेआउट_गुरुत्व:** यह लेआउट का गुरुत्व निर्धारित करता है।



एंड्रॉइड ग्रिडव्यू

एंड्रॉइड ग्रिड व्यू लेआउट हमारे पास मौजूद सबसे महत्वपूर्ण लेआउट में से एक है। यह हमें डेटा को ग्रिड के रूप में दिखाने में मदद करता है ताकि हम एक बार में कई इमेज या आइकन दिखा सकें। हम गैलरी से ऑडियो, वीडियो या इमेज दिखाने के लिए एप्लिकेशन में इसका उपयोग करते हैं।

ग्रिड मूल रूप से **2-आयामी दृश्य** में आइटम की सूची दिखाता है।

यदि आइटम की सूची स्क्रीन में फ़िट नहीं हो रही है, तो यह स्वचालित रूप से स्कॉल करने योग्य हो जाती है, हमें स्कॉल व्यू या इसके साथ कुछ और की आवश्यकता नहीं है। ग्रिड को कंटेनर में मौजूद एंड्रॉइड ग्रिड व्यू की मदद से बनाया जाता है। कंटेनर की सूची से ग्रिड को खींचा जा सकता है और स्क्रीन पर गिराया जा सकता है। लेआउट को आइकन या छवियों से भरने के लिए हमें Adapter का उपयोग करने की आवश्यकता है। एक Adapter केवल स्रोत से सामग्री लेकर ग्रिड व्यू में डेटा सम्मिलित करता है।



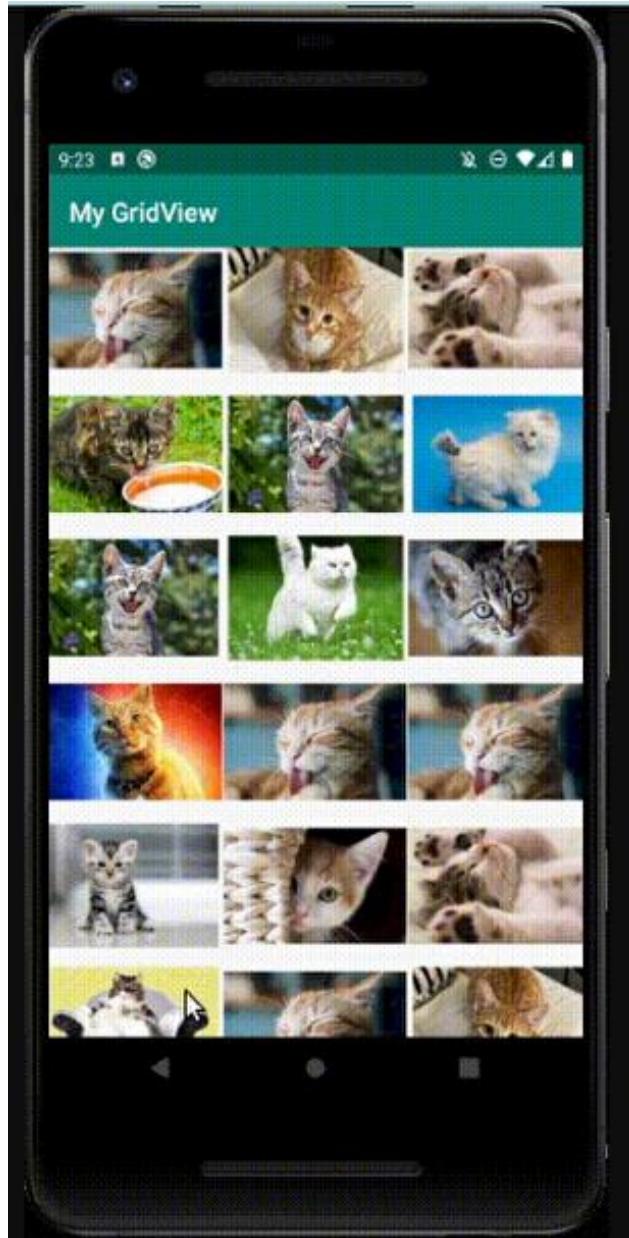
Android GridView की विशेषताएँ

जब हम Android ग्रिडव्यू को परिभाषित करते हैं, तो हम इसे अपनी ज़रूरतों और इच्छा के अनुसार कस्टमाइज़ भी कर सकते हैं। ग्रिड लेआउट को कस्टमाइज़ करने के लिए हम निम्नलिखित विशेषताओं का उपयोग करते हैं :

- android: id** – आवश्यकता पड़ने पर इसे चुनने के लिए ग्रिड को विशिष्ट रूप से पहचाना जाना चाहिए। इस उद्देश्य के लिए, इस विशेषता का उपयोग करके प्रत्येक ग्रिड के लिए id परिभाषित किया जाता है।
- android: gravity** - यह विशेषता ग्रिड व्यू का संरेखण निर्धारित करती है। हम ग्रिड को बाएँ, दाएँ, नीचे, आदि का उपयोग करके संरेखित कर सकते हैं।

- android: columnWidth** – इस विशेषता का उपयोग कॉलम की चौड़ाई को dp, sp, px, in या mm का उपयोग करके एक निश्चित आकार में सेट करने के लिए किया जाता है।
- android: स्ट्रेचमोड** - स्ट्रेचमोड विशेषता का उपयोग सेल में खाली स्थान को फैलाने और भरने के लिए किया जाता है, यदि कोई हो। इसके द्वारा रखे जाने वाले मान हैं - कोई नहीं, स्पेसिंगविड्थ, कॉलमविड्थ, स्पेसिंगविड्थयूनिफॉर्म।
- android: क्षैतिज स्पेसिंग** - यह विशेषता कॉलम के बीच की दूरी को परिभाषित करती है। स्पेसिंग dp, sp, px, in या mm में हो सकती है।

- **android: numColumns** – यह विशेषता उन आइकन या छवियों की संख्या निर्धारित करती है जिन्हें एक कॉलम में व्यवस्थित किया जा सकता है।
- **android: height** – यह विशेषता Android GridView की ऊँचाई dp, sp, px, in या mm में सेट करती है।
- **android: width** – यह विशेषता ग्रिड व्यू की चौड़ाई dp, sp, px, in या mm में सेट करती है।
- **android: verticalSpacing** – यह विशेषता पंक्तियों के बीच की दूरी को परिभाषित करती है। यह दूरी dp, sp, px, in या mm में हो सकती है।



CardView



CardView एंड्रॉयड में एक नया विजेट है जिसका उपयोग किसी भी प्रकार के डेटा को एक विशिष्ट ऊंचाई के साथ एक गोल कोने वाला लेआउट प्रदान करके प्रदर्शित करने के लिए किया जा सकता है। CardView वह दृश्य है जो एक दूसरे के ऊपर दृश्य प्रदर्शित कर सकता है। CardView का मुख्य उपयोग यह है कि यह UI डिज़ाइन को एक समृद्ध अनुभव और रूप देने में मदद करता है। इस विजेट को कई अलग-अलग Android ऐप्स में आसानी से देखा जा सकता है। CardView का उपयोग लिस्टव्यू में या रीसाइक्लर व्यू के अंदर आइटम बनाने के लिए किया जा सकता है। CardView के बारे में सबसे अच्छी बात यह है कि यह Framelayout को बढ़ाता है और इसे Android के सभी प्लेटफॉर्म पर प्रदर्शित किया जा सकता है।

कार्डव्यू की कुछ महत्वपूर्ण विशेषताएं हैं:

1. **cardBackgroundColor** : कार्ड का बैकग्राउंड-कलर सेट करने के लिए उपयोग किया जाता है।
2. **cardElevation** : कार्ड की ऊँचाई (किसी ऊँचे स्थान या अधिक महत्वपूर्ण स्थिति में जाने की प्रक्रिया) को परिभाषित करता है। इसका मान बहुत बड़ा नहीं होना चाहिए अन्यथा डिजाइन अच्छा नहीं लग सकता है।
3. **cardCornerRadius** : यह कार्ड के कोनों के चारों ओर त्रिज्या सेट करता है। इस विशेषता का मान जितना अधिक होगा कार्ड में किनारे उतने ही गोलाकार दिखाई देंगे।
4. **cardUseCompactPadding** : इसके दो मान सत्य और असत्य हैं। डिफॉल्ट रूप से, कार्डव्यू स्क्रीन के ऊपरी बाएँ कोने में (0,0) पर सेट होता है। और यदि यह विशेषता सत्य पर सेट है, तो कार्ड अपने लिए पैडिंग सेट कर देगा ताकि हमारा UI अच्छा दिखे।

RecyclerView

[RecyclerView](#) , [ListView](#) और [GridView](#) का अधिक लचीला और उन्नत संस्करण है। RecyclerView का उपयोग बड़े डेटा सेट को सीमित विंडो प्रदान करने के लिए किया जाता है, जिसका अर्थ है कि इसका उपयोग बड़ी मात्रा में डेटा प्रदर्शित करने के लिए किया जाता है जिसे सीमित संख्या में व्यू बनाए रखकर बहुत कुशलता से स्क्रॉल किया जा सकता है। RecyclerView में हम डेटा प्रदान करते हैं और परिभाषित करते हैं कि प्रत्येक आइटम कैसा दिखता है, और RecyclerView लाइब्रेरी गतिशील रूप से सामग्री बनाती है जब इसकी आवश्यकता होती है।



How RecyclerView Works?

- **RecyclerView** एक व्यूग्रुप है जिसमें आपके डेटा से संबंधित व्यू शामिल होते हैं। यह खुद एक व्यू है, इसलिए इसे लेआउट फ़ाइल में उसी तरह जोड़ा जाता है जैसे कोई अन्य UI तत्व जोड़ा जाता है।
- **ViewHolder** ऑब्जेक्ट का उपयोग सूची में प्रत्येक individual element को परिभाषित करने के लिए किया जाता है। व्यू होल्डर में बनाए जाने पर कुछ भी नहीं होता है, RecyclerView डेटा को इससे बांधता है। ViewHolder को `RecyclerView.ViewHolder` को विस्तारित करके परिभाषित किया जाता है।

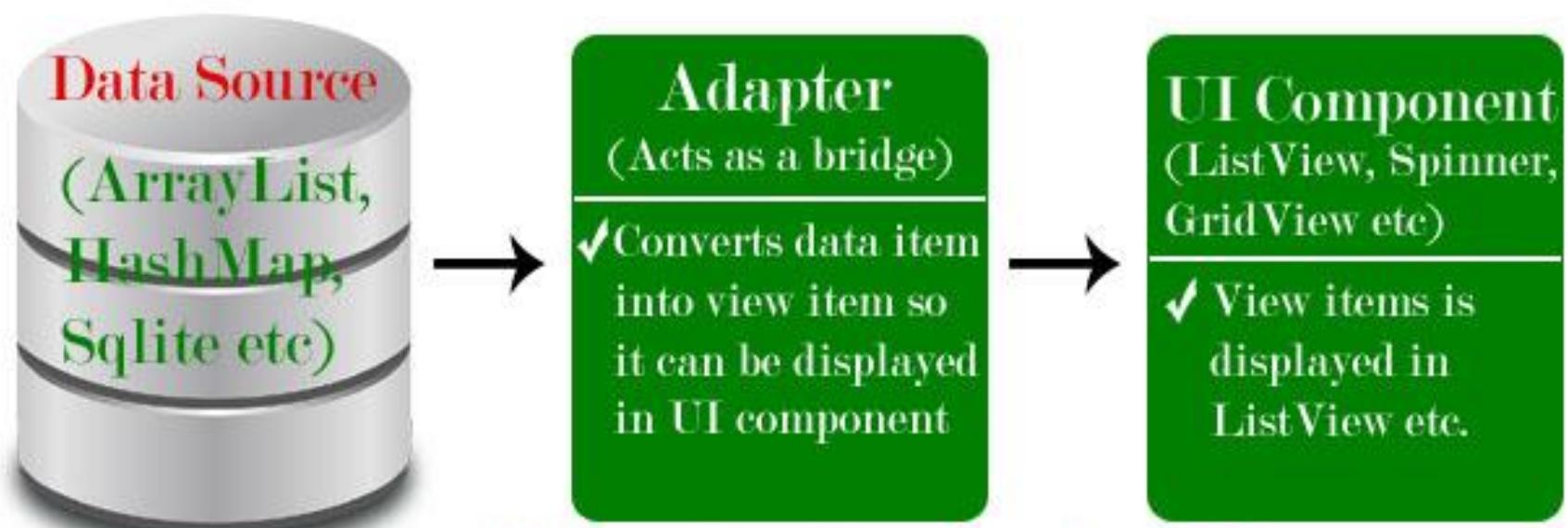
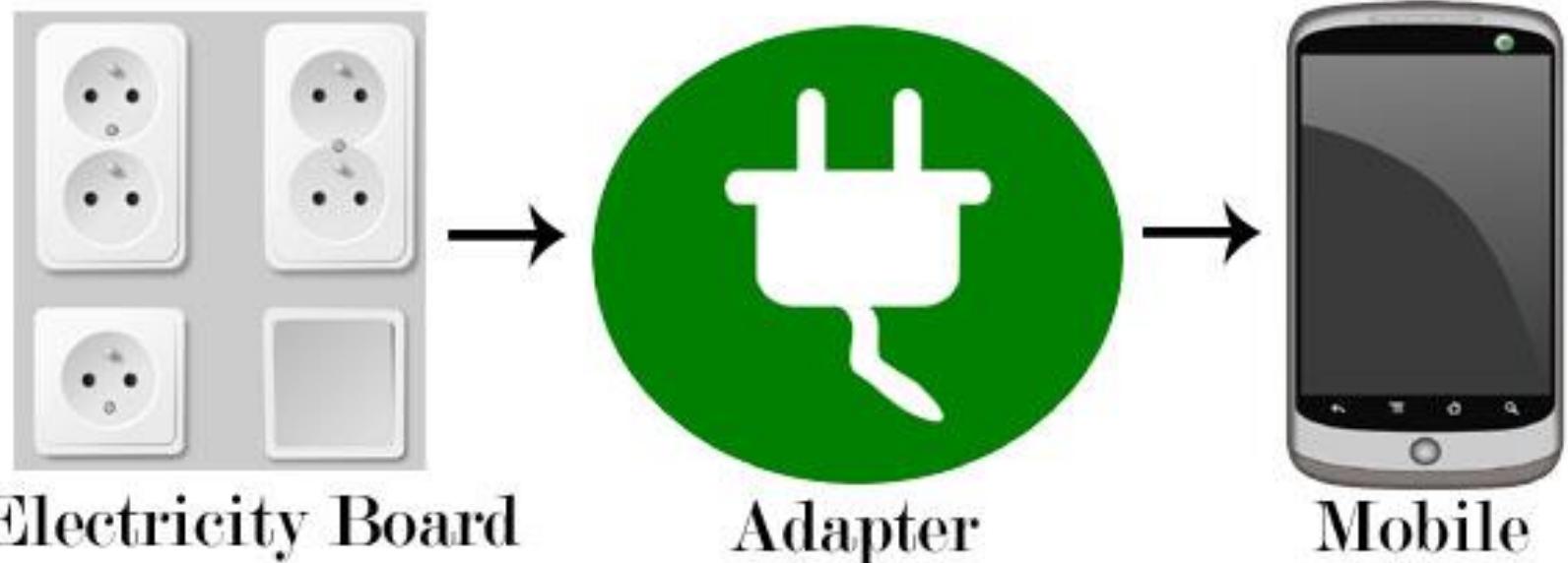
How RecyclerView Works?

- **Adapters** का उपयोग डेटा को व्यू से जोड़ने के लिए किया जाता है। RecyclerView व्यू का अनुरोध करता है, और Adapter में विधियों को कॉल करके व्यू को उनके डेटा से जोड़ता है। Adapter को `RecyclerView.Adapter` को विस्तारित करके परिभाषित किया जा सकता है।
- **LayoutManager** सूची में अलग-अलग तत्वों को व्यवस्थित करता है। इसमें उन सभी दृश्यों का संदर्भ शामिल होता है जो प्रविष्टि के डेटा से भरे होते हैं।

Adapter

एंड्रॉयड में, Adapter यूआई घटक और डेटा स्रोत के बीच एक पुल है जो हमें यूआई घटक में डेटा भरने में मदद करता है। यह डेटा को होल्ड करता है और डेटा को Adapter व्यू में भेजता है फिर व्यू Adapter व्यू से डेटा ले सकता है और डेटा को अलग-अलग व्यू जैसे लिस्ट व्यू, ग्रिड व्यू, स्पिनर आदि पर दिखाता है। व्यू में अधिक अनुकूलन के लिए हम बेस Adapter या कस्टम Adapter का उपयोग करते हैं।

किसी सूची या ग्रिड में डेटा भरने के लिए हमें Adapter को लागू करने की आवश्यकता है। Adapter UI घटक और डेटा स्रोत के बीच एक पुल की तरह काम करता है। यहाँ डेटा स्रोत वह स्रोत है जहाँ से हम डेटा प्राप्त करते हैं और UI घटक सूची या ग्रिड आइटम हैं जिसमें हम उस डेटा को प्रदर्शित करना चाहते हैं।



Adapter In Android

Adapters In Android

एंड्रॉइड में कुछ सामान्यतः प्रयुक्त Adapter हैं जिनका उपयोग यूआई घटकों में डेटा भरने के लिए किया जाता है।

1. BaseAdapter - यह अन्य सभी एडाप्टरों के लिए मूल Adapter है
2. ArrayAdapter - इसका उपयोग तब किया जाता है जब हमारे पास एकल आइटम की एक सूची होती है जो एक सरणी द्वारा समर्थित होती है
3. Custom ArrayAdapter - इसका उपयोग तब किया जाता है जब हमें कस्टम सूची प्रदर्शित करने की आवश्यकता होती है

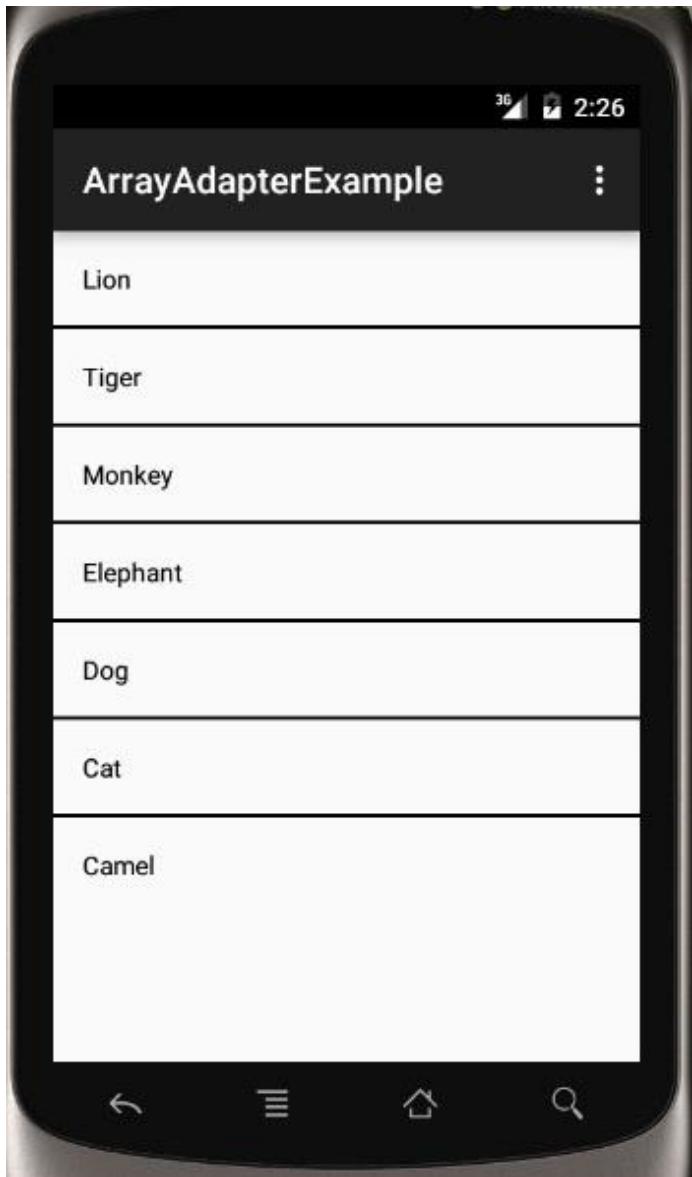
Adapters In Android

4. SimpleAdapter - यह आपके XML फ़ाइल में परिभाषित दृश्यों में स्थैतिक डेटा को मैप करने के लिए एक आसान Adapter है
5. Custom SimpleAdapter- इसका उपयोग तब किया जाता है जब हमें एक अनुकूलित सूची प्रदर्शित करने की आवश्यकता होती है और सूची या ग्रिड के चाइल्ड आइटम तक पहुंचने की आवश्यकता होती है

1. BaseAdapter

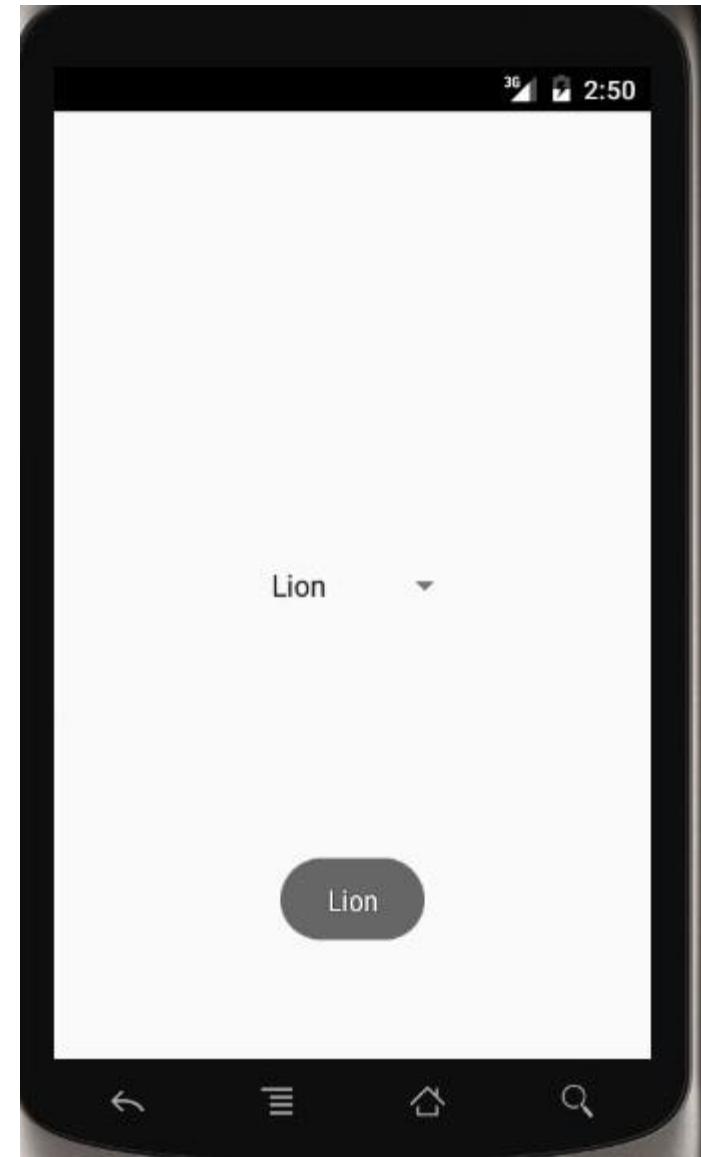
बेसAdapter एक Adapter के सामान्य implementation का एक common base class है जिसका उपयोग [ListView](#) , [GridView](#) , [Spinner](#) आदि में किया जा सकता है। जब भी हमें [ListView](#) में कस्टमाइज़्ड सूची या [GridView](#) में कस्टमाइज़्ड ग्रिड की आवश्यकता होती है तो हम अपना स्वयं का Adapter बनाते हैं और उसमें बेस Adapter का विस्तार करते हैं। कस्टम सूची आइटम प्रदर्शित करने के लिए कस्टम Adapter बनाने के लिए बेस Adapter का विस्तार किया जा सकता है। ArrayAdapter भी BaseAdapter का एक implementation है।

List View



1. BaseAdapter

Spinner



2. ArrayAdapter

जब भी हमारे पास list of single items होती है जो Array द्वारा समर्थित होती है, तो हम ArrayAdapter का उपयोग कर सकते हैं। उदाहरण के लिए, फ़ोन संपर्कों, देशों या नामों की सूची।

ArrayAdapter(Context context, int resource, int textViewResourceId, T[] objects)

3. Custom ArrayAdapter

ArrayAdapter भी BaseAdapter का ही एक implementation है, इसलिए यदि हम और अधिक customization चाहते हैं तो हम एक कस्टम एडाप्टर बना सकते हैं और उसमें ArrayAdapter को विस्तारित कर सकते हैं। चूंकि ऐसे एडाप्टर BaseAdapter का ही एक implementation है, इसलिए हम अपने कस्टम एडाप्टर में BaseAdapter के सभी फ़ंक्शन को ओवरराइड कर सकते हैं।

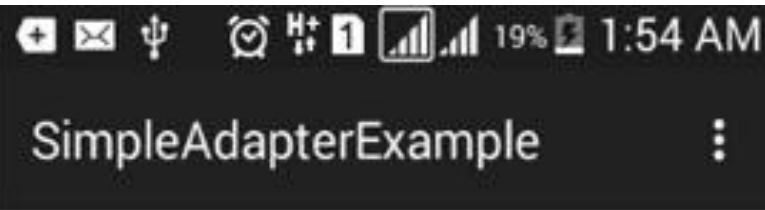
4. SimpleAdapter

एंड्रॉइड में SimpleAdapter एक आसान एडाप्टर है जो स्टेटिक डेटा को [XML](#) फ़ाइल (लेआउट) में defined views में मैप करने के लिए है। एंड्रॉइड में हम किसी list में डेटा बैकिंग को [मैप्स](#) (यानी हैशमैप या अन्य) के ArrayList के रूप में निर्दिष्ट कर सकते हैं। ArrayList में प्रत्येक entry list की एक row के अनुरूप होती है।

मैप में प्रत्येक पंक्ति के लिए डेटा होता है। यहाँ हम एक [XML](#) फ़ाइल (कस्टम सूची आइटम फ़ाइल) भी define करते हैं जो पंक्ति को प्रदर्शित करने के लिए उपयोग किए जाने वाले views को परिभाषित करती है, और मैप में कुंजियों से विशिष्ट दृश्यों तक मैपिंग करती है।

5. Custom SimpleAdapter

जब भी हमें कोई कस्टम सूची बनानी होती है तो हमें कस्टम एडाप्टर को लागू करने की आवश्यकता होती है। जैसा कि हमने पहले चर्चा की है, ArrayAdapter का उपयोग तब किया जाता है जब हमारे पास Array द्वारा समर्थित एकल आइटम की सूची होती है। इसलिए यदि हमें ListView या GridView में customization की आवश्यकता है तो हमें सरल एडाप्टर को लागू करने की आवश्यकता है, लेकिन जब हमें सूची या ग्रिड आइटम में अधिक customization की आवश्यकता होती है, जहां हमारे पास सूची आइटम में कई view होते हैं और फिर हमें किसी विशेष view पर क्लिक या कोई अन्य इवेंट करना होता है, तो हमें एक कस्टम एडाप्टर को लागू करने की आवश्यकता होती है जो हमारी आवश्यकता को पूरा करता है और लागू करना काफी आसान है।



Apple



Banana



Litchi



Mango



PineApple



Apple

Android Dialogs

Dialog एक छोटी सी window होती है जिससे यूज़र कोई decision ले सकता है या कुछ information input कर सकता है। Dialog पूरी screen को cover नहीं करता है।

Android dialogs 3 तरह के होते हैं -

1. Alert dialog – ये एक simple alert dialog होता है। इस तरह के dialog में आप title शो कर सकते हैं, 3 बटन add कर सकते हैं और items की list भी शो कर सकते हैं जिन्हे यूज़र select कर सकता है। ये dialog क्रिएट करने के लिए आपको AlertDialog क्लास को extend करना पड़ेगा।

2. Date picker dialog – इस तरह के dialog से आप यूज़र को एक window शो कर सकते हैं जिससे यूज़र date select कर सकता है। इस तरह का dialog क्रिएट करने के लिए आपको DatePickerDialog क्लास को extend करना होगा।

3. Time picker dialog – इस तरह के dialog से यूज़र time select कर सकता है। इस तरह का dialog क्रिएट करने के लिए आपको TimePickerDialog क्लास को extend करना होगा।

Android Toasts

Android toast एक simple popup मैसेज होता है जो कुछ देर के लिए स्क्रीन पर appear होता है और automatically disappear हो जाता है। उदाहरण के लिए जब आप अपने device से कोई मैसेज भेजते हैं, तो आपको एक toast शो होता है, जिसमे लिखा होता है “Message sent” और वो कुछ देर बाद automatically disappear हो जाता है।

Android toasts की size उतनी ही होती है, जितना बड़ा आपका मैसेज होता है। Current activity visible होती है, आप इसके साथ interact कर सकते हैं।

Methods Used with Android Toasts

makeText()

ये method एक toast create करने के काम में आता है. इस मेथड को आप Toast क्लास के object पर कॉल करते हैं। इस method में आप current application का context, आपका text message और toast की duration पास करते हैं। इसका structure नीचे दिया जा रहा है।

```
public static Toast makeText(context-of-Application, text-Message, duration-of-Message);
```

makeText() method का example नीचे दिया जा रहा है।

```
Toast.makeText(getApplicationContext(), "Welcome to JE Classes", Toast.LENGTH_SHORT);
```

show()

ये मेथड toast को display करता है। इस मेथड को आप Toast क्लास के object पर call करते हैं।

```
toastObject.show();
```

setGravity()

इस मेथड से आप अपने toast मैसेज को अपनी application में मन चाही position पर सेट कर सकते हैं।

```
toastObject.setGravity(200,50,200);
```

Creating Android Toasts

- 1.Android में toasts क्रिएट करने के लिए सबसे पहले Toast क्लास का ऑब्जेक्ट क्रिएट करेंगे।
- 2.इसके बाद Toast क्लास के object पर makeText() मेथड कॉल कीजिये और उसमे current application का context, text message और duration पास कीजिये।
- 3.इसके बाद toast object पर setGravity() मेथड कॉल करके toast मैसेज की position को सेट कीजिये।
- 4.इसके बाद toast को show करने के लिए toast object पर show() मेथड कॉल कीजिये।

SimpleToast_Example

CLICK

This a simple toast message

एंड्रॉयड में थ्रेड

Android में थ्रेड important concepts है। थ्रेड एक हल्का सब-प्रोसेस है जो हमें यूजर इंटरफ़ेस (UI) को बाधित किए बिना बैकग्राउंड ऑपरेशन करने का एक तरीका प्रदान करता है। जब कोई ऐप लॉन्च होता है, तो यह एक सिंगल थ्रेड बनाता है जिसमें सभी ऐप components डिफ़ॉल्ट रूप से चलेंगे। रनटाइम सिस्टम द्वारा बनाया गया थ्रेड मुख्य थ्रेड के रूप में जाना जाता है।

मुख्य थ्रेड की प्राथमिक भूमिका UI में इवेंट हैंडलिंग और views के साथ interaction के संदर्भ में UI को संभालना है। यदि कोई ऐसा कार्य है जो समय लेने वाला है और वह कार्य मुख्य थ्रेड पर चलाया जाता है, तो यह पूरा होने तक अन्य कार्यों को रोक देगा, जिसके परिणामस्वरूप ऑपरेटिंग सिस्टम द्वारा उपयोगकर्ता को “Application is unresponsive” चेतावनी प्रदर्शित हो सकती है। इसलिए हमें ऐसे कार्यों और कुछ अन्य कार्यों के लिए अलग-अलग थ्रेड की आवश्यकता होती है।

सभी थ्रेडिंग घटक दो मूल श्रेणियों में से एक के अंतर्गत आते हैं।

•**The fragment or activity attached threads:** थ्रेड की यह श्रेणी गतिविधि / फ्रैगमेंट के जीवनचक्र से जुड़ी होती है और गतिविधि/फ्रैगमेंट के नष्ट होते ही ये समाप्त हो जाती हैं।

थ्रेड घटक:

AsyncTask , लोडर।

•**The fragment or activity not attached threads:** इस प्रकार के धागे उस गतिविधि या खंड के जीवनकाल से आगे भी चलते रह सकते हैं, जिससे वे उत्पन्न हुए थे।

थ्रेडिंग घटक: Service, Intent Service.

दो थ्रेडिंग घटकों के लिए, एंड्रॉइड मोबाइल विकास में पांच प्रकार के थ्रेड का उपयोग किया जाता है।

मैन थ्रेड: जब हम एंड्रॉइड पर अपना ऐप लॉन्च करते हैं, तो यह execution का पहला थ्रेड बनाता है जिसे "मैन थ्रेड" कहा जाता है। एंड्रॉइड UI Toolkit से components के बीच संचार और उनके उचित यूआई विजेट्स को events का dispatch मुख्य थ्रेड द्वारा नियंत्रित किया जाता है। हमें नेटवर्क संचालन, डेटाबेस कॉल और मुख्य थ्रेड में कुछ components को लोड करने से बचना चाहिए। क्योंकि execute होने पर मुख्य थ्रेड को सिंक्रोनस रूप से कॉल किया जाता है, इसका मतलब है कि प्रदर्शन पूरा होने तक user interface पूरी तरह से unresponsive रहेगा।

UI Thread

Android में प्रत्येक ऐप का अपना थ्रेड होता है जो UI ऑब्जेक्ट्स, जैसे कि व्यू ऑब्जेक्ट्स को चलाने के लिए जिम्मेदार होता है। ऐसे थ्रेड को UI थ्रेड के रूप में जाना जाता है। UI थ्रेड हमारे ऐप के लिए execution का मुख्य थ्रेड है क्योंकि यह वह जगह है जहाँ अधिकांश ऐप कोड चलाया जाता है। UI थ्रेड वह जगह है जहाँ हमारे सभी ऐप component (जैसे activities, services, content providers, and broadcast receivers) बनाए जाते हैं। यह थ्रेड हमारे कार्यों को उनके बैकग्राउंड कार्य को perform करने और फिर परिणामों को बिटमैप जैसे UI तत्वों में ले जाने की अनुमति देता है। हम जो कार्य थ्रेड पूल से थ्रेड पर चलाते हैं वे हमारे UI थ्रेड पर नहीं चलते हैं, इसलिए उनके पास UI ऑब्जेक्ट्स तक पहुँच नहीं होगी। डेटा UI थ्रेड पर चलने वाले हैंडलर का उपयोग करके बैकग्राउंड थ्रेड से UI थ्रेड पर जाता है।

Worker Thread

वर्कर थ्रेड एक बैकग्राउंड थ्रेड है। वर्कर थ्रेड को UI थ्रेड जैसे थ्रेड के अलावा अलग से बनाया जाता है। जैसा कि हम नियमों से जानते हैं, हम UI थ्रेड को ब्लॉक नहीं कर सकते हैं, इसलिए यहीं पर वर्कर थ्रेड काम आता है क्योंकि हम चाइल्ड प्रोसेस और टास्क को चलाने के लिए उनका इस्तेमाल कर सकते हैं।

Any Thread

Any Thread में, एनोटेट विधि को किसी भी थ्रेड से कॉल किया जा सकता है। यदि एनोटेट element एक क्लास है, तो क्लास में सभी methods को किसी भी थ्रेड से कॉल किया जा सकता है।

```
@Target([
    AnnotationTarget.FUNCTION, AnnotationTarget.PROPERTY_GETTER,
    AnnotationTarget.PROPERTY_SETTER, AnnotationTarget.CONSTRUCTOR,
    AnnotationTarget.CLASS, AnnotationTarget.FILE,
    AnnotationTarget.VALUE_PARAMETER
])
```

```
class AnyThread
```

Binder Thread

बाइंडर थ्रेड सेवा के एक अलग थ्रेड का प्रतिनिधित्व करता है। बाइंडर एक mechanism है जो inter-process communication प्रदान करता है। बाइंडर थ्रेड का उपयोग inter-process communication के साथ सेवा बाइंडिंग में किया जाता है। यह concept मुख्य रूप से एंड्रॉइड इंटरफ़ेस डेफिनेशन लैंग्वेज (AIDL) द्वारा परिभाषित इंटरफ़ेस के साथ सेवा कॉल से संबंधित है।

THREADS

Main Thread

aka UI Thread

Avoid long operations

All user input + output

Background Thread

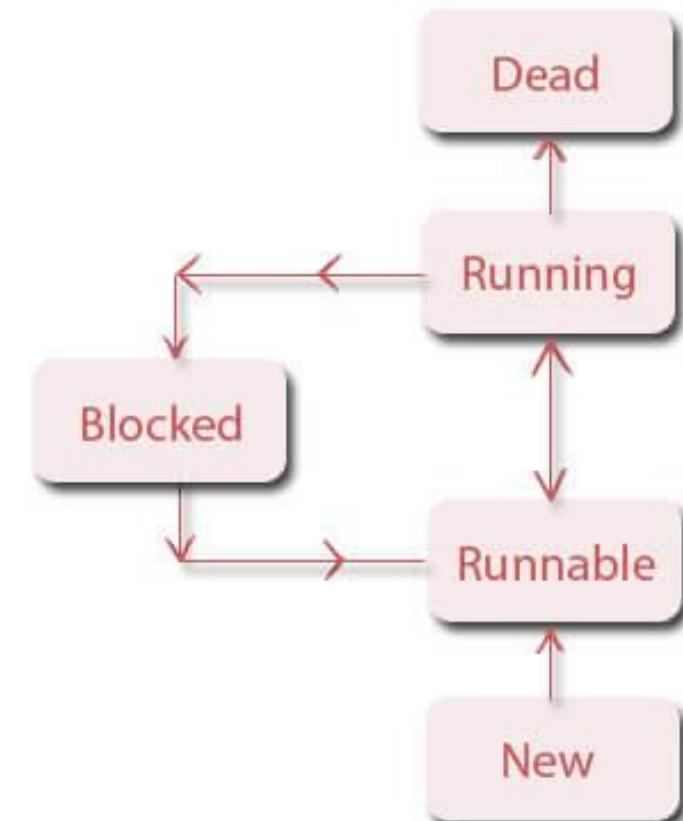
for long-running work



Life Cycle of a Thread

थ्रेड की लाइफ साइकिल में पांच stages होती हैं

- **New** – एक थ्रेड अपना लाइफ साइकिल new state में शुरू करता है. यह इस state में तब तक रहता है जब तक start() method को call नहीं किया जाता.
- **Runnable** – जब थ्रेड को start() method के द्वारा start कर दिया जाता है तो यह runnable हो जाता है अर्थात् यह runnable state में चला जाता है.
- **Running** – एक थ्रेड running state में होता है यदि scheduler उसे select करता है.
- **Waiting** – एक थ्रेड तब waiting state में होता है जब वह एक task को परफॉर्म करने के लिए दूसरे task के लिए wait करता है.
- **Terminated** – जब thread अपने task पूरा कर देता है तो वह terminated state में चला जाता है.



Lifecycle of a thread

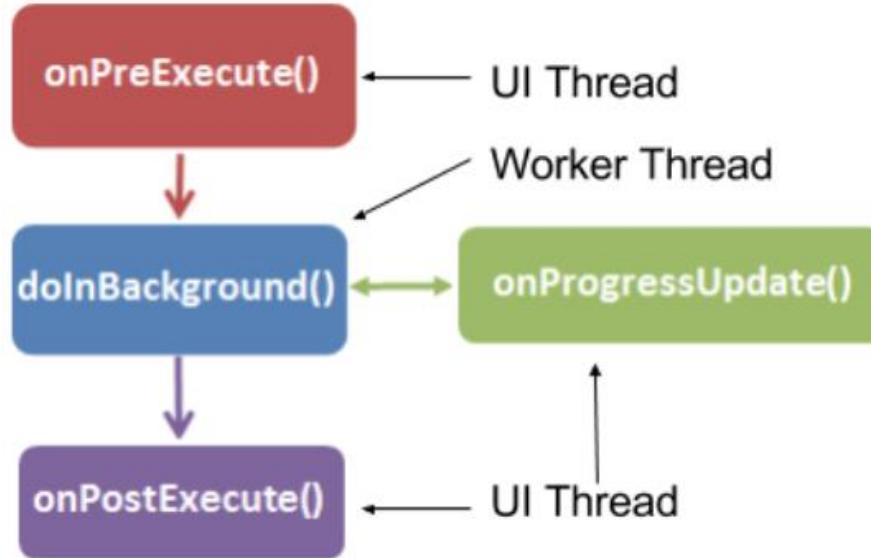
AsyncTask

AsyncTask (एसिंक्रोनस टास्क) एक एंड्रॉइड क्लास है जो बैकग्राउंड थ्रेड पर ऑपरेशन करना आसान बनाता है और थ्रेड्स और हैंडलर्स में स्वयं हेरफेर किए बिना यूजर इंटरफ़ेस (UI)/मुख्य थ्रेड पर परिणाम प्रकाशित करता है।

एंड्रॉयड में, एक अलग थ्रेड से UI को अपडेट करना एक अक्सर सामना किया जाने वाला परिदृश्य है। यह इतना आम है कि एंड्रॉयड ऐसे परिदृश्यों के लिए उपयोग करने के लिए तैयार API प्रदान करता है।

AsyncTask मूल रूप से एक API है।

ASYNCTASK की विधियाँ



AsyncTask में चार विभिन्न प्रकार की विधियाँ हैं जो यह सुनिश्चित करती हैं कि आप लंबे कार्यों को उस तरीके से करने में सक्षम हैं जो आप मुख्य थ्रेड में नहीं करते हैं।

ये विधियाँ इस प्रकार हैं:

> **onPreExecute()** - विधि का यह भाग मुख्य थ्रेड पर निष्पादित होता है। बैकग्राउंड ऑपरेशन करने से पहले आपको उपयोगकर्ता को स्क्रीन पर कोई प्रोग्रेस बार या एनीमेशन जैसी कोई चीज़ दिखानी चाहिए।

>**doinBackground(Params)** - इस विधि का उपयोग बैकग्राउंड थ्रेड पर बैकग्राउंड ऑपरेशन करने के लिए किया जाता है। doInBackground() विधि में ऑपरेशन किसी भी मुख्य थ्रेड गतिविधि या फ्रैगमेंट को स्पर्श नहीं करना चाहिए।

>**onProgressUpdate(Progress...)** - यह विधि doInBackground विधि से प्रगति अपडेट प्राप्त करती है और UI थ्रेड पर चलती है। इस विधि का उपयोग UI में प्रगति बार प्रदर्शित करने के लिए किया जाता है।

>**onPostExecute(Result)** - इस विधि में आप पृष्ठभूमि ऑपरेशन परिणाम के UI को अपडेट कर सकते हैं।

Android में AsyncTask के सामान्य प्रकार:

एसिंक्रोनस कार्य द्वारा उपयोग किये जाने वाले तीन प्रकार निम्नलिखित हैं:

`AsyncTask <TypeOfVarArgParams, ProgressValue, ResultValue>`

1. TypeOfVarArgParams: Params execution के समय कार्य को भेजे जाने वाले पैरामीटर का प्रकार है।

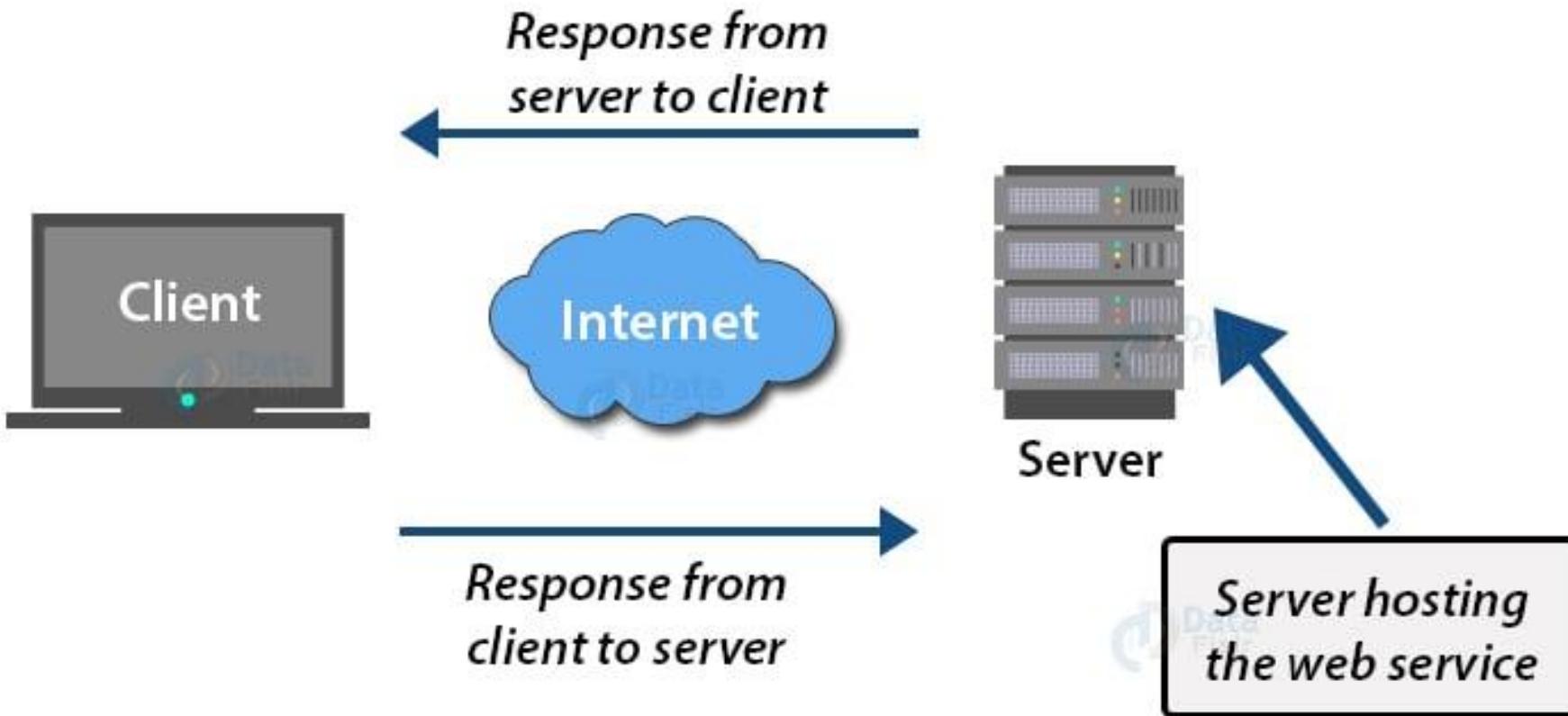
2. ProgressValue: Progress background computation के दौरान प्रकाशित progress units का प्रकार है।

3. ResultValue: ResultValue background computation के परिणाम का प्रकार है।

एंड्रॉइड Web Services क्या हैं?

वेब सेवा मूल रूप से अंतर-संचालन क्षमता प्रदान करने के लिए आवश्यक है, अर्थात् विभिन्न Applications को जोड़ना। यह विभिन्न ऐप्स को एक-दूसरे के साथ संवाद करने और आपस में डेटा और सेवाओं को साझा करने की अनुमति देता है। Web Services सभी प्रकार के क्लाइंट Applications के लिए हर प्रकार के ऐप सर्वर पर फ़ंक्शन को लागू करने के लिए एक मानक प्रदान करती हैं।

How Web Servers Work?



Android Web Services Components

a. Publisher

publisher को Service provider के रूप में समझा जा सकता है। publisher वेब सेवा बनाने और उसे ग्राहकों के लिए उपलब्ध कराने के लिए जिम्मेदार होता है।

b. Subscriber

सब्सक्राइबर और कुछ नहीं बल्कि service requester है। service requester वह है जिसे वेब सेवा से संपर्क करने की आवश्यकता है। क्लाइंट एप्लिकेशन क्लाइंट एप्लिकेशन के माध्यम से संपर्क करेगा। यह क्लाइंट एप्लिकेशन .Net या किसी भी भाषा पर आधारित हो सकता है।

c. Broker

यहाँ ब्रोकर वह एप्लीकेशन है जो UDDI तक access प्रदान करता है। UDDI का मतलब है User descriptive, discovery and integration। यह क्लाइंट एप्लीकेशन को web services का सटीक पता लगाने में सक्षम बनाता है।

ये सेवाएं इस प्रकार हैं:

a. Publish

प्रकाशक वेब सेवाओं को प्रकाशित करने का मतलब है ब्रोकर को इसके अस्तित्व के बारे में सूचित करना। यह ब्रोकर के इंटरफ़ेस का उपयोग करके किया जाता है ताकि ग्राहकों के लिए इसे आसानी से सुलभ बनाया जा सके।

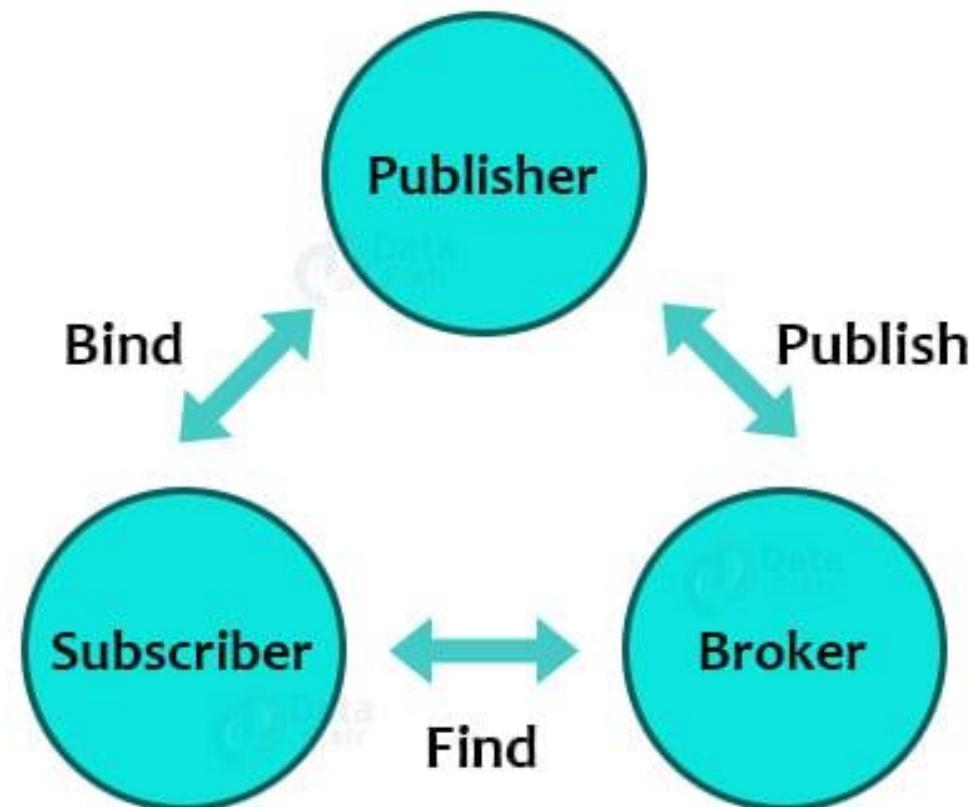
b. Subscribe

ग्राहक प्रकाशित web services को आसानी से ढूँढ़ने के लिए ब्रोकर से परामर्श करेगा।

c. Bind

एक बार ब्रोकर से वेब सेवाओं के बारे में जानकारी प्राप्त हो जाने पर, ग्राहक वेब सेवा को बाँध सकता है।

Relation between Publisher, Subscriber & Broker



Characteristics of Web Services in Android

1. Web Services XML-आधारित हैं। वे इसका उपयोग data representational layer और transportational layer पर करते हैं क्योंकि यह नेटवर्किंग, ऑपरेटिंग सिस्टम या यहां तक कि प्लेटफ़ॉर्म बाइंडिंग को हटा देता है। ये सेवाएँ अपने core level पर अत्यधिक interoperable हैं।
2. Web Services loosely coupled होती हैं। इसका मतलब है कि उपभोक्ता Web Services और वेब सेवा प्रदाता सीधे एक दूसरे से बंधे नहीं होते हैं।
3. वेब सेवाओं में सिंक्रोनस या एसिंक्रोनस होने की क्षमता होती है। यहाँ सिंक्रोनस का मतलब क्लाइंट को सेवा के execution से बांधना है। दूसरी ओर, एसिंक्रोनस का मतलब क्लाइंट को पहले एक सेवा शुरू करने और बाद में अन्य कार्यों को execute करने की अनुमति देना है।

Characteristics of Web Services in Android

4. Web Services **Remote Procedure Calls** का समर्थन करती हैं। रिमोट प्रोसीजर कॉल को अक्सर RPC के रूप में संदर्भित किया जा सकता है। ये RPC क्लाइंट को XML का उपयोग करके रिमोट ऑब्जेक्ट पर विभिन्न फ़ंक्शन, विधियाँ और सेवाएँ लागू करने देते हैं।
5. वेब सेवाओं में Document exchange का समर्थन है। वास्तव में, XML में डेटा के साथ-साथ जटिल दस्तावेज़ों को प्रस्तुत करने का एक बहुत ही सामान्य तरीका है। इसके साथ ही, इसमें इन दस्तावेज़ों को प्रस्तुत करने के विभिन्न तरीके हैं।

Types of Web Services

1. XML-RPC

In XML-RPC, RPC stands for remote procedure calls. यह इंटरनेट पर कई तरह के उपकरणों के बीच डेटा के आदान-प्रदान के लिए XML आधारित प्रोटोकॉल है।

2. UDDI

UDDI stands for Universal Descriptive, discovery, and integration. यह एक XML-आधारित मानक है जिसका उपयोग नई वेब सेवाओं के विवरण, प्रकाशन और खोज के लिए किया जाता है।

3. SOAP

SOAP here stands for Simple object access protocol. यह एक XML आधारित वेब सेवा प्रोटोकॉल है जिसका उपयोग HTTP (हाइपरटेक्स्ट ट्रांसफर प्रोटोकॉल) या SMTP (सिंपल मैसेज ट्रांसफर प्रोटोकॉल) पर डेटा या documents के आदान-प्रदान के लिए किया जाता है। यह अलग-अलग सिस्टम पर काम करने वाली independent processes के communication की अनुमति देता है।

4. REST

Here, REST is Representational State Transfer. यह डिवाइस और इंटरनेट के बीच संचार और कनेक्टिविटी प्रदान करता है।

Advantages of Web Services

1. वेब सेवाएँ विभिन्न Applications के बीच interoperability को सक्षम बनाती हैं।
2. वेब सेवाओं का उपयोग करने का एक बहुत ही महत्वपूर्ण लाभ Reusability है।
3. Web services , Applications और organizations के भीतर और उनके बीच तेज़ communications प्रदान करती हैं।
4. वे विभिन्न Applications के बीच संचार को सक्षम करने के लिए एक quality industry-standard protocol का उपयोग करते हैं।
5. वे वेब सेवाओं को लागू करने के लिए कम लागत वाले इंटरनेट के उपयोग को सक्षम करने के लिए HTTP पर SOAP का उपयोग करते हैं।
6. वेब सेवाओं को मानक इंटरनेट तकनीकों पर तैनात किया जाता है।
7. वे हमें इंटरनेट पर मौजूदा कोड के कार्यों को उजागर करने की अनुमति देते हैं।

Android Web Services Limitations

1. वेब सेवाएँ ब्राउज़र से एक्सेस नहीं करती हैं।
2. वे emerging Web development का लाभ नहीं उठाती हैं।
3. वेब सेवाओं द्वारा उपयोग किया जाने वाला HTTP प्रोटोकॉल विश्वसनीय नहीं है और असुरक्षित है।

Looper



लूपर

क्लास का उपयोग थ्रेड के लिए संदेश लूप चलाने के लिए किया जाता है। डिफॉल्ट रूप से थ्रेड के साथ कोई संदेश लूप संबंध नहीं होता है; इसे बनाने के लिए, उस थ्रेड को कॉल करें जो लूप को चलाना है, और फिर लूप बंद होने तक संदेशों को प्रोसेस करने के लिए कहें।

लूपर वह डाकिया है जो संदेश या कार्य को हैंडलर तक पहुंचाता है

हैंडलर

हैंडलर आपको थ्रेड से जुड़े Runnable ऑब्जेक्ट भेजने और प्रोसेस करने की अनुमति देता है। प्रत्येक हैंडलर इंस्टेंस एक एकल थ्रेड और उस थ्रेड की संदेश कतार से जुड़ा होता है।

हैंडलर वह व्यक्ति होता है जो नियुक्त डाकिया के साथ पत्र भेजता और प्राप्त करता है। पत्र दो प्रकार के हो सकते हैं। एक सिर्फ़ डेटा है, जिसे हम संदेश कह सकते हैं, दूसरा निष्पादन है, जिसे हम रननेबल कहते हैं।

हैंडलर.भेजें(संदेश);

हैंडलर.पोस्ट(रननेबल)

कभी-कभी, हैंडलर डेटा संदेश प्राप्त करने वाला रिसीवर भी हो सकता है। और यह अपने हैंडलमैसेज(मैसेज एमएसजी) फंक्शन में संदेश को संभाल सकता है।

संदेश

एक संदेश को परिभाषित करता है जिसमें एक विवरण और मनमाना डेटा ऑब्जेक्ट होता है जिसे भेजा जा सकता है। इस ऑब्जेक्ट में दो अतिरिक्त `int` फ़ील्ड और एक अतिरिक्त ऑब्जेक्ट फ़ील्ड होती है जो आपको कई मामलों में आवंटन नहीं करने देती है।

SERVICES

- Service एक facility होती है जिससे आप android system को बता सकते हैं कि आप कोई काम background में करते रहना चाहते हैं।
- Service एक facility होती है जिससे एक एप्लीकेशन अपनी कुछ functionality दूसरी applications को expose कर सकती है।

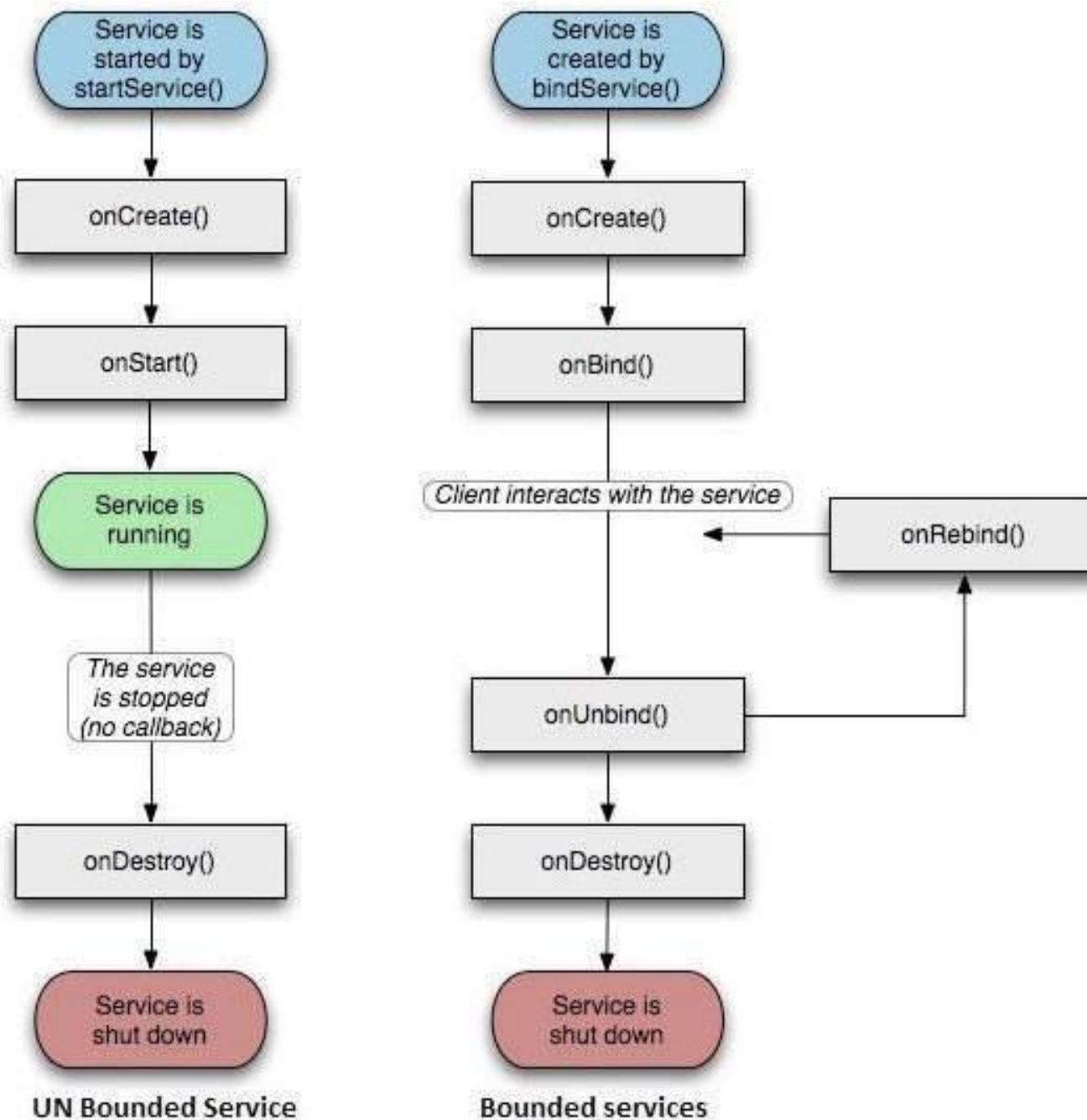
Services दो प्रकार की होती हैं -

- १- LOCAL services – access from within the application
- २- Remote services – accessed remotely from other applications running on the same device.

एक service की 2 State हो सकती है-

- **Started** - एक service started service कहलाती है जब कोई component startService() मेथड कॉल करता है। एक बार स्टार्ट होने के बाद service बैकग्राउंड में execute होती रहेगी चाहे इसे क्रिएट करने वाला component destroy हो जाये। जब operation कम्पलीट हो जाता है तो service automatically stop हो जाती है।
- **Bounded** - एक एप्लीकेशन bound होती है जब कोई component bindService() मेथड कॉल करके service को bind करता है। एक bound service client server interface प्रोवाइड करती है जिससे component service के साथ interact कर सकता है।

Life Cycle of services



•**onStartCommand()** - Android system इस मेथड को तब कॉल करता है जब कोई दूसरा component जैसे की कोई activity startService() कॉल करके service को start करने के लिए request करता है। जैसे ही ये मेथड कॉल होता है service स्टार्ट हो जाती है और बैकग्राउंड में execute होती रहती है। अगर आप इस मेथड को implement करते हैं तो stopService() मेथड कॉल करके service को स्टॉप करना आपकी जिम्मेदारी होती है।

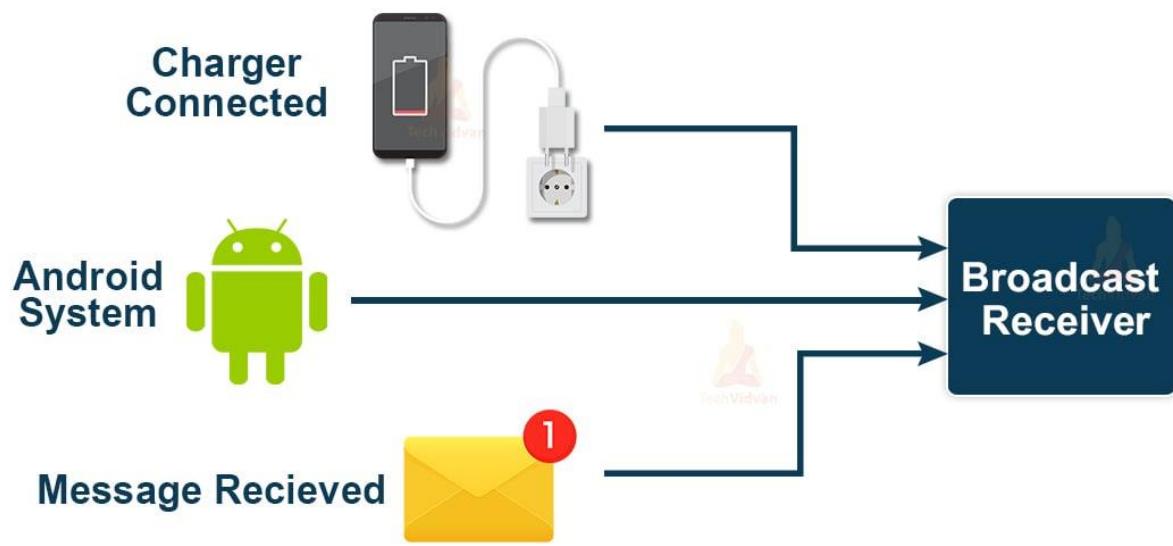
•**onBind()** - Android system ये मेथड तब कॉल करता है जब कोई component bindService() मेथड कॉल करके service के साथ bind होना चाहता है। जब आप इस मेथड को implement करे तो आपको एक interface प्रोवाइड करना चाहिए जिससे clients service से communicate कर सके।

- **onUnbind ()** - सिस्टम इस **method** को तब कॉल करता है जब सभी क्लाइंट सेवा द्वारा प्रकाशित किसी विशेष इंटरफ़ेस से डिस्कनेक्ट हो जाते हैं।
- **onRebind ()** - जब नए **client service** से जुड़ जाते हैं , तो **system** इस **method** को कॉल करता है
- **onCreate()** - **Android system** इस **method** को तब कॉल करता है जब **service** फर्स्ट टाइम क्रिएट होती है। ये मेथड सिर्फ एक बार ही कॉल होता है।
- **onDestroy()** - **Android system** इस मेथड को तब कॉल करता है जब **service** अपना काम पूरा कर लेती है और **destroy** होती है।

Broadcast Receiver

Broadcast receivers वो android components होते हैं जिनकी मदद से आप दूसरी applications में और system में generate होने वाले events के बारे में notification पा सकते हैं।

किसी भी event के बारे में जानकारी पाने के लिए आपकी application को उस event के लिए खुद को register करवाना पड़ता है। कोई भी application ऐसा एक broadcast receiver क्रिएट कर सकती है। जब भी कोई event generate होता है तो android system उन सभी application को notify करता है जिन्होंने event के लिए खुद को रजिस्टर किया था।



```
public class MyReceiver extends BroadcastReceiver
{
    public void onReceive(context,intent)
    {
    }
}
```

निम्नलिखित दो महत्वपूर्ण स्टेप्स के द्वारा BroadcastReceiver बनाया जाता है –

- Creating the Broadcast Receiver
- Registering Broadcast Receiver

Broadcast receivers क्रिएट करने के लिए आपको BroadcastReceiver क्लास को extend करना होगा। BroadcastReceiver क्लास के onReceive() मेथड को आप अपने हिसाब से implement कर सकते हैं। onReceive() method में आप डिफाइन करते हैं कि event receive करने के बाद आप क्या करना चाहते हैं।

Broadcast receiver क्रिएट करने के बाद आपको intent के लिए register भी करना होता है। ऐसा आप AndroidManifest.xml फाइल में <receiver> element को define करके करते हैं।

Classes of Broadcast in Android

- 1.Ordered Broadcasts
- 2.Normal Broadcasts

1. Ordered Broadcasts

ऑर्डर किए गए broadcasts उचित क्रम में किए जाते हैं क्योंकि उन्हें कुछ priority दी जाती है। priority android:priority विशेषता का उपयोग करके broadcasts को सौंपी जाती है। priority की मदद से, सबसे अधिक priority वाला broadcast पहले execute होगा। यदि दो प्रसारणों की priority समान पाई जाती है तो कोई क्रम नहीं अपनाया जाएगा। ऑर्डर किए गए प्रसारणों को सिंक्रोनस broadcasts भी कहा जाता है।

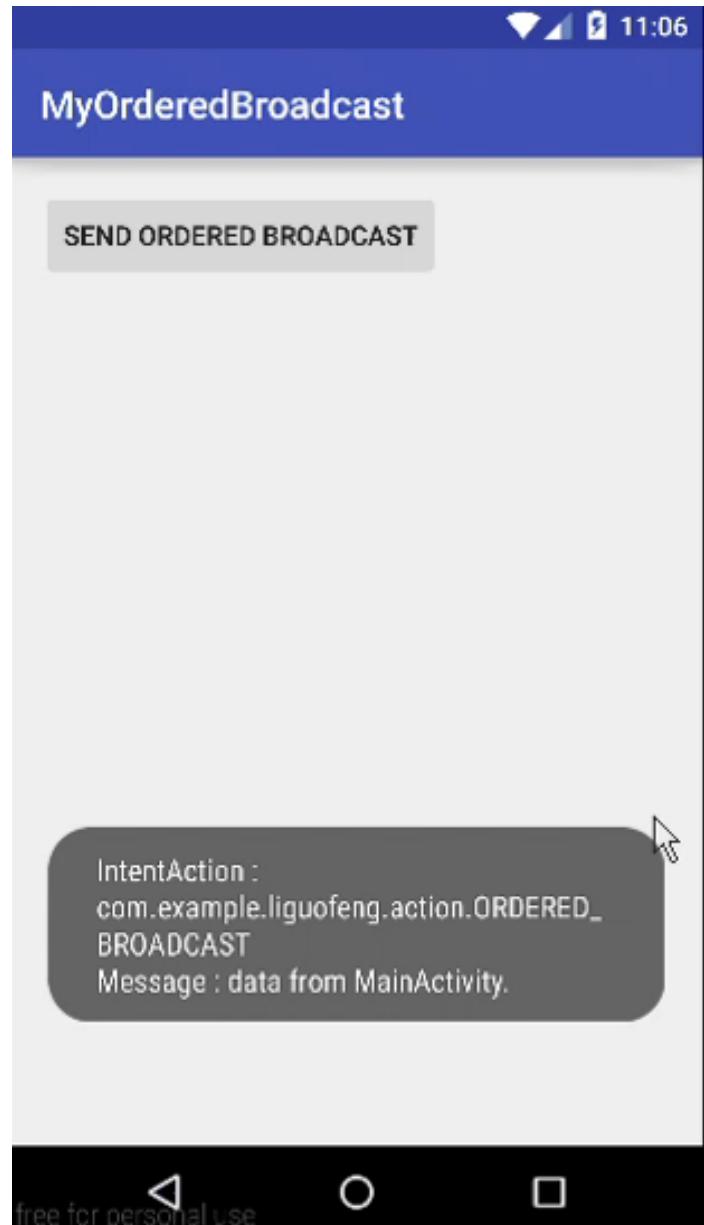
इसमें एक broadcasts एक समय में केवल एक रिसीवर को ही दिया जाता है। जब कोई रिसीवर broadcasts प्राप्त करता है तो यह रिसीवर पर निर्भर करता है कि वह broadcasts को पास करे या रोके। यदि रिसीवर चाहे तो वह broadcasts को अगले रिसीवर को पास कर देता है अन्यथा broadcasts अगले रिसीवर तक नहीं पहुँचता।

Normal Broadcasts

Normal broadcasts asynchronous और unordered होते हैं। ये रिसीवर अव्यवस्थित रूप से या कभी-कभी एक साथ चलते हैं। इसलिए वे efficient होते हैं, लेकिन परिणामों का पूरा उपयोग नहीं कर पाते। ये broadcasts **Context:sendBroadcast** का उपयोग करके भेजे जाते हैं।

Normal broadcasts में, यह संभव है कि ओवरहेड से बचने के लिए सिस्टम एक बार में केवल एक ही प्रसारण भेजे। यह प्रसारण को निरस्त भी कर सकता है, लेकिन API को छोड़कर।

एप्लिकेशन से प्राप्त होने वाली Notification को भी म्यूट किया जा सकता है।



Android notification

Notifications आपके device में top पर battery और signal icons के पास show होते हैं। आपने देखा होगा जैसे ही आपके device पर कोई mail या text message आता है तो top पर एक message icon शो होता है। ये messages notifications कहलाते हैं।

जब भी कोई notification आता है तो सबसे पहले उसका icon show होता है। जैसे कि कोई mail आया तो Gmail का icon शो होगा। इसके बाद यदि यूज़र उस notification के बारे में detail जानना चाहता है तो वो notification drawer को open करता है।

ज्यादातर devices में notification drawer जब यूज़र top से bottom की तरफ swipe करता है तो open हो जाता है। Notification drawer को यूज़र कभी भी देख सकता है। इसमें कुछ दूसरी settings भी रहती है जैसे कि WiFi, Bluetooth, screen brightness आदि होती हैं।

आप चाहे तो notification के माध्यम से कोई action भी ले सकते हैं। जैसे कि जब यूज़र notification पर click करे तो आपकी application open हो जाये।

Contents of Android Notification

- **Icon** – ये एक छोटा सा icon होता है जो notification से related एप्लीकेशन को represent करता है
- **Title** – ये notification का title होता है जो कि notification का purpose show करता है।
- **Detail text** – ये detailed text message होता है।

Creating a Notification

एक notification को create करने के लिए सबसे पहले आप NotificationCompat.Builder का class का object create करते हैं। यह एक public static class है। जो आसानी से notifications क्रिएट करने के लिए ये एक builder क्लास है। इसमें आप current activity का reference पास करते हैं।

इसके बाद आप इस builder क्लास के object के माध्यम से notification के contents सेट करेंगे। Icon set करने के लिए setSmallIcon() method कॉल करेंगे और इसमें icon का एड्रेस पास करेंगे। Notification का title सेट करने के लिए आप setContentTitle() method कॉल करते हैं और इसमें title string पास करते हैं। Detailed message set करने के लिए setContentText() method कॉल करते हैं और इसमें text string पास करते हैं।

इसके बाद आप NotificationCompat.Builder क्लास के object पर build() method कॉल करते हैं। ये method एक notification return करता है। जो notification ये method return करता है तो वो उन्हीं contents के साथ loaded होता है जो आपने set किये थे।

सबसे आखिर में notification को issue करने के लिए आप build method ने जो notification object return किया जाता है उसे notify() method में पास करते हैं। इस मेथड को NotificationManager क्लास के object पर कॉल करते हैं।

```
NotificationCompat.Builder ncb = new NotificationCompat.Builder(this); // creating  
builder class object  
  
ncb.setSmallIcon(R.drawable.yricon); // setting icon  
  
ncb.setContentTitle("your notification title"); //setting title  
  
ncb.setContextText("your notification message");// setting text  
  
Notification noti = ncb.build(); //creating notification  
  
NotificationManager nm = new NotificationManager();  
  
nm.notify(noti); //showing notification
```

4:23 PM

Thursday, October 2



The Big Meeting

4:15 – 5:15 PM
Big Conference Room



MAP



EMAIL GUES...



New Google+ notifications 3:44 PM
Earl Liibyrd: Added you back 2



Screenshot captured. 4:22 PM
Touch to view your screenshot.



Keep photos & videos backed.. 4:22 PM
Touch to get free private storage on Google



3 new messages 4:11 PM
kumlatar.swankatranami@gmail.com 3



3 new messages 2:42 PM
(754) 263-8267, 456



Android Storage System

Android system data को स्टोर करने के लिए बहुत से options provide करता है। आप कौनसा option choose करते हैं ये आप पर depend करता है। यदि आपका data primitive है तो आप shared preferences use कर सकते हैं। यदि आपका data private है तो आप उसे internal storage में store कर सकते हैं। यदि आपकी application के लिए database में डेटा store करना जरूरी है तो आप SQLite database में डेटा स्टोर कर सकते हैं। और यदि आप चाहे तो network connection के द्वारा किसी cloud storage में भी अपना data स्टोर कर सकते हैं।

Shared Preferences

SharedPreferences एक क्लास होती है। ये क्लास आपको एक ढांचा provide करती है। इस क्लास में आप key value के pair में केवल primitive values (int , boolean, char, float, strings) store कर सकते हैं। जो डेटा आप shared preferences में store करते हैं वो different user sessions में भी available रहता है।

SharedPreferences क्लास का object प्राप्त करने के लिए आपके पास 2 options होते हैं। एक तो

आप getSharedPreferences() method को call कर सकते हैं। ये method आपको तब ही use करना चाहिए जब आप बहुत सी preferences files create करना चाहते हैं। दूसरा option ये है की आप getPreferences() method को कॉल करे इस method के द्वारा आप सिर्फ एक ही preference file create कर सकते हैं।

सबसे पहले आप SharedPreferences क्लास का object क्रिएट करेंगे। इसमें आप variable का नाम और initial value पास करते हैं।

```
SharedPreferences myObj = new SharedPreferences(variable_name,  
initial-value);
```

SharedPreferences क्लास का object क्रिएट करने के बाद value store करने के लिए आपको SharedPreferences.Editor क्लास का ऑब्जेक्ट क्रिएट करना होगा। ऐसा आप SharedPreferences क्लास के object पर edit() मेथड को कॉल करके कर सकते हैं।

```
SharedPreferences.Editor ediObj = myObj.edit();
```

अलग अलग तरह की values store करने के लिए अलग अलग methods हैं। जैसे की putBoolean() और putString() आदि। इन methods को आप editor object पर कॉल करते हैं।

```
ediObj.putString("name",value);
```

इसके बाद आपको values को commit करना होता है।

```
ediObj.commit();
```

Shared Preferences in Android



METHODS OF SHARED PREFERENCES

1.contains(String key) : इस विधि का उपयोग यह जांचने के लिए किया जाता है कि वरीयताओं में कोई वरीयता शामिल है या नहीं।

2.edit() : इस विधि का उपयोग इन प्राथमिकताओं के लिए एक नया संपादक बनाने के लिए किया जाता है, जिसके माध्यम से आप प्राथमिकताओं में डेटा में संशोधन कर सकते हैं और उन परिवर्तनों को SharedPreferences ऑब्जेक्ट में वापस भेज सकते हैं।

3.getAll() : इस विधि का उपयोग प्राथमिकताओं से सभी मानों को पुनः प्राप्त करने के लिए किया जाता है।

4. **getBoolean(String key, boolean defValue)** : इस विधि का उपयोग वरीयताओं से बूलियन मान प्राप्त करने के लिए किया जाता है।
5. **getFloat(String key, float defValue)** : इस विधि का उपयोग प्राथमिकताओं से फ्लोट मान प्राप्त करने के लिए किया जाता है।
6. **getInt(String key, int defValue)** : इस विधि का उपयोग प्राथमिकताओं से int मान प्राप्त करने के लिए किया जाता है।

7. **getLong(String key, long defValue)** : इस विधि का उपयोग वरीयताओं से लंबा मान प्राप्त करने के लिए किया जाता है।
8. **getString(String key, String defValue)** : इस विधि का उपयोग प्राथमिकताओं से स्ट्रिंग मान प्राप्त करने के लिए किया जाता है।
9. **getStringSet(String key, Set defValues)** : इस विधि का उपयोग वरीयताओं से स्ट्रिंग मानों का एक सेट प्राप्त करने के लिए किया जाता है।

10.registerOnSharedPreferencechangeListener(SharedPreferences.OnSharedPreferencechangeListener listener) : इस विधि का उपयोग किसी वरीयता में परिवर्तन होने पर कॉलबैक को पंजीकृत करने के लिए किया जाता है।

11.unregisterOnSharedPreferencechangeListener(SharedPreferences.OnSharedPreferencechangeListener listener) : इस विधि का उपयोग पिछले कॉलबैक को अपंजीकृत करने के लिए किया जाता है।

Internal Storage in Android

internal storage पर डेटा save और load करना उस एप्लिकेशन के लिए private है जिसे अन्य एप्लिकेशन द्वारा एक्सेस नहीं किया जा सकता है। जब ऐप को uninstall किया जाता है तो उस ऐप द्वारा internal में stored डेटा हटा दिया जाता है।

किसी भी file को internal storage में save करने के लिए आपको openFileOutput() method को कॉल करना होगा। ये method FileOutputStream class का object return करता है। FileOutputStream क्लास data को file के through output करने के लिए यूज़ की जाती है। ये क्लास एक stream को represent करती है। openFileOutput() method में आप file का नाम और mode pass करते हैं।

File के 4 mode होते हैं।

- **MODE_PRIVATE** – ये default mode होता है। इस mode में file private रहती है।
- **MODE_APPEND** – यदि पास किये गए नाम की file पहले से है तो नया डेटा उसी file के data के आखिर में add हो जाता है।
- **MODE_WORLD_READABLE** – इस तरह की files create करना security के point of view से बहुत risky होता है। इस तरह की फाइल को कोई भी दूसरी application access कर सकती है।
- **MODE_WORLD_WRITABLE** – इस तरह की file को कोई भी दूसरी application modify कर सकती है।

FileOutputStream का object पाने के बाद आप उस पर write() मेथड को कॉल करते हैं और file को save कर देते हैं। इसके बाद आप close method call करके output stream close कर देते हैं।

```
String name="yrName";
FileOutputStream obj = openFileOutput(fileName,Context.MODE_PRIVATE);

obj.write(name.getBytes()); // write data in form of byte only.
obj.close(); // closing stream
```

इसी प्रकार internal storage में से files को read करने के लिए आपको openFileInput() method को कॉल करना होगा। ये method FileInputStream क्लास का object return करता है। इस method में आप file का नाम पास करते हैं। इसके बाद आप FileInputStream के object पर read() मेथड कॉल करके file को read करते हैं।

```
FileInputStream obj = openFileInput(fileName);
obj.read();
obj.close();
```



External Storage

हर android device में ये capability होती है कि आप उसमे external memory add कर सकते है। ये memory removable भी हो सकती है जैसे कि SD card आदि और non removable भी हो सकती है। कई बार internal memory की limit के कारण भी इस external memory को यूज़ किया जाता है। External storage में save होने वाली सभी classes world readable mode की होती है। और जब आप अपने device को PC से connect करते है तो इन files को कोई भी modify कर सकता है।

External Storage

यदि आपकी application external storage में files save करना चाहती है तो इसके लिए उसको 2 तरह की system permissions की आवश्यकता होगी।

- READ_EXTERNAL_STORAGE** – ये permission external storage से files को read करने के लिए होती है।
- WRITE_EXTERNAL_STORAGE** – ये permission external storage में files save करने के लिए होती है।

`<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>`

External storage में कोई भी काम करने से पहले आपको memory का media status चेक कर लेना चाहिए। ये भी हो सकता है कि external storage में कोई file ही न हो या सिर्फ readable files हो। External storage का status check करने के लिए आप getExternalStorageState() method को कॉल कर सकते हैं। ये method आपको files की अलग अलग states return करता है जिन से आप यूज़र को notify कर सकते हैं।

यदि आप files को public रखना चाहते हैं तो आप उन्हें public directories में save कर सकते हैं जैसे की Music, Pictures आदि। External storage में files save करना बहुत ही आसान है। इसके लिए आप एक file ऑब्जेक्ट create करते हैं और उसमे file का नाम और उस public directory का नाम भी पास करते हैं।

```
File myFile = newFile(DIRECTORY_PUBLIC, fileName);
```

एंड्रॉइड - SQLite डेटाबेस

SQLite एक ओपनसोर्स SQL डेटाबेस है जो डिवाइस पर टेक्स्ट फ़ाइल में डेटा संग्रहीत करता है। Android में बिल्ट इन SQLite डेटाबेस implementation आता है।

SQLite सभी रिलेशनल डेटाबेस सुविधाओं का समर्थन करता है। इस डेटाबेस तक पहुँचने के लिए, आपको इसके लिए JDBC, ODBC आदि जैसे किसी भी प्रकार के कनेक्शन स्थापित करने की आवश्यकता नहीं है।

Android SQLite Database



Name	_id	Place
Priya	00001	Indore
Riya	00011	Delhi
Sandy	00111	Mumbai
Raj	01111	Pune

SQLite क्या है?

SQLite मूल रूप से एक रिलेशनल डेटाबेस मैनेजमेंट सिस्टम (RDBMS) है, जो SQL की तरह ही है। यह एक ओपन-सोर्स इन-प्रोसेस लाइब्रेरी है जो self-contained, serverless, zero-configuration, और एक ट्रांजेक्शनल SQL डेटाबेस इंजन है। यहाँ, zero-configuration का मतलब है कि अन्य डेटाबेस प्रबंधन प्रणालियों के विपरीत, इसे डिवाइस पर कॉन्फिगर करने की आवश्यकता नहीं है।

Features of Android SQLite

- Full-featured वाला SQL implementation , साथ ही विभिन्न उन्नत क्षमताएं जैसे partial indexing, JSON, और कुछ अन्य।
- बहुत सरल और उपयोग में आसान एपीआई।
- कभी-कभी इसका execution बहुत तेज होता है, और यह direct file system Input-Output से भी अधिक तेज होता है।
- आत्मनिर्भर क्योंकि इसकी कोई बाहरी निर्भरता नहीं है।
- SQLite में लेनदेन में ACID गुण भी होता है जो कि - Atomicity, Consistency, Isolation, and Durability है।
- क्रॉस-प्लेटफॉर्म क्योंकि यह एंड्रॉइड, आईओएस, लिनक्स, मैक, विंडोज, VxWorks, Solaris का समर्थन करता है। इसे अन्य सिस्टम में पोर्ट करना आसान है।

Features of Android SQLite

- एक स्टैडअलोन कमांड-लाइन इंटरफ़ेस क्लाइंट के साथ आता है।
- पूर्णतः एक ही क्रॉस-प्लेटफॉर्म पर संग्रहीत।
- पूर्णतः ANSI-C में लिखा गया।
- यह बहुत हल्का और छोटा है, क्योंकि इसे 400Kbs से भी कम में कॉन्फ़िगर किया जा सकता है। यदि वैकल्पिक सुविधाओं को छोड़कर कॉन्फ़िगर किया जाए, तो यह 250Kbs से भी कम का उपयोग करता है।

एंड्रॉयड में SQLite

तो, हम जानते हैं कि SQLite एक ओपन-सोर्स RDBMS है जिसका उपयोग rows और columns के रूप में संग्रहीत डेटाबेस पर संचालन करने के लिए किया जाता है। SQLite, Android द्वारा highly supported है; वास्तव में, Android SQLite के built-in डेटाबेस implementation के साथ आता है। यह प्रत्येक Android डेटाबेस पर उपलब्ध है और scalability, centralization, concurrency, और control पर जोर देता है। यह applications और devices के लिए local level पर storage प्रदान करने का प्रयास करता है। यह अत्यधिक economical, efficient, reliable, and independent है। यह बहुत सरल है और इसे क्लाइंट/सर्वर डेटाबेस से तुलना करने की आवश्यकता नहीं है।

1. SQLite निम्नलिखित तीन प्रकार के डेटा का समर्थन करता है:

- **Text Type** – to store strings or character type data.
- **Integer Type** – to store the integer data type.
- **Real Type** – to store long values.

2. Android एप्लिकेशन में SQLite का उपयोग करने के लिए, हम android.database.sqlite पैकेज का उपयोग करते हैं। इस पैकेज में SQLite का उपयोग करने के लिए सभी API शामिल हैं।

3. **SQLiteOpenHelper** एक क्लास है जो डेटाबेस बनाने और उसे प्रबंधित करने के लिए उपयोगी है। [SQLiteOpenHelper क्लास](#) के दो कंस्ट्रक्टर हैं:

•**SQLiteOpenHelper(Context context, String name, SQLiteDatabase.CursorFactory factory, int version_no)** – यह डेटाबेस बनाने, खोलने और प्रबंधित करने के लिए ऑब्जेक्ट बनाता है।

•**SQLiteOpenHelper(Context context, String name, SQLiteDatabase.CursorFactory factory, int version_no, DatabaseErrorHandler errorHandler)** – इसका ऑब्जेक्ट डेटाबेस बनाने, खोलने और प्रबंधित करने के साथ-साथ त्रुटि हैंडलर को निर्दिष्ट करता है।

What is CRUD?

CRUD is nothing but an abbreviation for the basic operations that we perform in any database. And the operations are

- **Create**
- **Read**
- **Update**
- **Delete**

The Database Structure

employees		
id	int	PK
name	varchar(200)	
department	varchar(200)	
joiningdate	datetime	
salary	double	

Creating the Table

```
CREATE TABLE employees (
    id INTEGER NOT NULL CONSTRAINT employees_pk PRIMARY KEY AUTOINCREMENT,
    name varchar(200) NOT NULL,
    department varchar(200) NOT NULL,
    joiningdate datetime NOT NULL,
    salary double NOT NULL
);
```

employees		
id	int	PK
name	varchar(200)	
department	varchar(200)	
joiningdate	datetime	
salary	double	

Creating a new Record

```
INSERT INTO employees (name, department,  
joiningdate, salary) VALUES ('Amit Chauhan',  
'Technical', '2017-09-30 10:00:00', '40000');
```

Reading All Existing Records

```
SELECT * FROM employees;
```

Reading Specific Record

```
SELECT * FROM employees WHERE id = 1;
```

* means selecting all the columns, if you want a specific column or multiple columns but not all you can write names of the columns like **SELECT name, department**.

Updating a Record

```
UPDATE employees SET name = 'Belal  
Haque', department = 'Research and  
Development', salary = '100000' WHERE id = 1;
```

Deleting a Record

```
DELETE FROM employees WHERE id = 1;
```

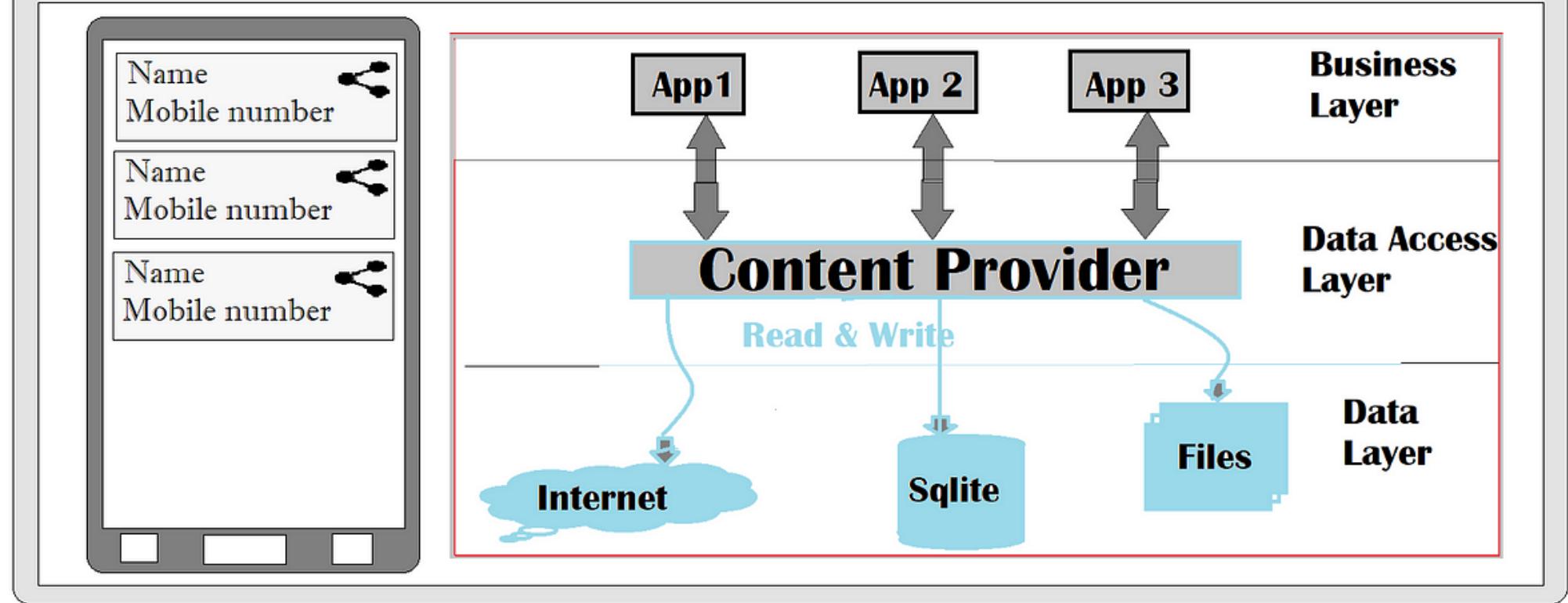
Content Provider

ऐसे Data, जिन्हें Device के Multiple Applications द्वारा Access किया जाना हो, को इन Applications हेतु Access करवाने के लिए जिस तरीके का प्रयोग किया जाता है, उसे **Content Provider** कहते हैं।

यानी Android Development Model हमें **Content Providers** के माध्यम से Data को इस प्रकार से Develop करने की सुविधा Provide करता है, ताकि उन्हें Device के अन्य Applications भी उपयोग में ले सकें। साथ ही हमारे Data पर इन Content Providers का पूरा Control भी होता है, जो इस बात को निश्चित करते हैं कि विभिन्न Applications हमारे Data को किस तरह से Access व Manipulate कर सकेंगे।

यहाँ Content Providers द्वारा Provide किए जाने वाले Content को हम किसी भी प्रकार का ऐसा Content मान सकते हैं, जिसे हमारे Application में Use किया जा सकता हो।

Content Provider in Android



उदाहरण के लिए हमारे Android Device का **Contacts App**, एक Content Provider का Best Example है क्योंकि हम हमारे किसी भी **SMS** या **Phone Dialer** Android App में इस Contact App के माध्यम से Stored **Contact List** को Access कर सकते हैं।

**Business
Layer**

Applications

**Data Access
Layer**

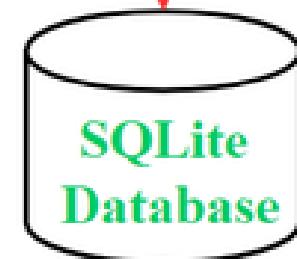
Content Provider

**Data
Layer**

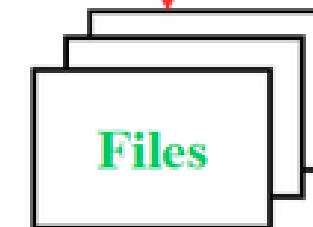


Internet

Read Write



**SQLite
Database**



Files

DG

Required methods

ContentProvider six abstract methods को परिभाषित करता है जिन्हें आप अपने concrete subclass के भाग के रूप में लागू करते हैं। इन सभी विधियों को छोड़कर onCreate() किसी क्लाइंट एप्लिकेशन द्वारा कॉल किया जाता है जो आपके content provider तक पहुँचने का प्रयास कर रहा है।

query()

अपने provider से डेटा प्राप्त करें। क्वेरी करने के लिए Table, लौटाने के लिए पंक्तियाँ और Column, तथा परिणाम का क्रम चुनने के लिए तर्कों का उपयोग करें।

insert()

अपने provider में एक नई Row डालें। destination Table का चयन करने और उपयोग करने के लिए Column मान प्राप्त करने के लिए तर्कों का उपयोग करें।

update()

अपने provider में मौजूदा पंक्तियों को अपडेट करें। अपडेट करने के लिए Table और पंक्तियों का चयन करने और अपडेट किए गए कॉलम मान प्राप्त करने के लिए तर्कों का उपयोग करें। अपडेट की गई पंक्तियों की संख्या लौटाएँ।

delete()

अपने provider से पंक्तियाँ हटाएँ। Table और पंक्तियों को हटाने के लिए तर्कों का उपयोग करें। हटाई गई पंक्तियों की संख्या लौटाएँ।

getType()

सामग्री URI के अनुरूप MIME प्रकार लौटाएँ। इस विधि का अधिक विस्तार से वर्णन सामग्री provider MIME प्रकार लागू करें अनुभाग में किया गया है।

onCreate()

अपने provider को आरंभ करें। एंड्रॉइड सिस्टम आपके provider को बनाने के तुरंत बाद इस विधि को कॉल करता है। आपका provider तब तक नहीं बनाया जाता जब तक कोई ContentResolver ऑब्जेक्ट इसे एक्सेस करने का प्रयास नहीं करता।