E Content of Mobile Application Development

Mobile Computing

Mobile Computing refers a technology that allows transmission of data, voice and video via a computer or any other wireless enabled device. It is free from having a connection with a fixed physical link. It facilitates the users to move from one physical location to another during communication.

Our Mobile communication tutorial includes all topics of mobile computing like its brief overview and history, evolution, classification, advantages and disadvantages, security issues, future trends etc.

The concept of Mobile Computing can be divided into three parts:

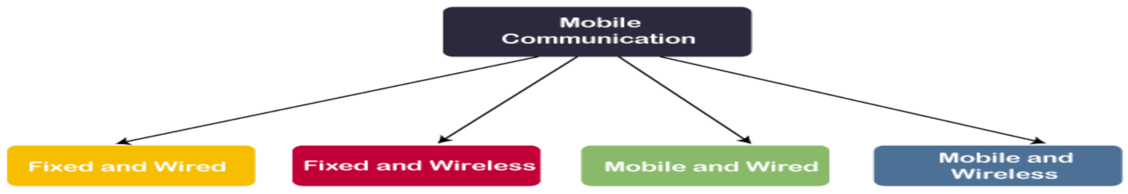
- Mobile Communication
- Mobile Hardware
- Mobile Software

Mobile Communication

Mobile Communication specifies a framework that is responsible for the working of mobile computing technology. In this case, mobile communication refers to an infrastructure that ensures seamless and reliable communication among wireless devices. This framework ensures the consistency and reliability of communication between wireless devices. The mobile communication framework consists of communication devices such as protocols, services, bandwidth, and portals necessary to facilitate and support the stated services. These devices are responsible for delivering a smooth communication process.

Mobile communication can be divided in the following four types:

1. Fixed and Wired
2. Fixed and Wireless
3. Mobile and Wired
4. Mobile and Wireless



Fixed and Wired: In Fixed and Wired configuration, the devices are fixed at a position, and they are connected through a physical link to communicate with other devices.

Mobile Hardware

Mobile hardware consists of mobile devices or device components that can be used to receive or access the service of mobility. Examples of mobile hardware can be smartphones, laptops, portable PCs, tablet PCs, Personal Digital Assistants, etc.

Mobile Software

Mobile software is a program that runs on mobile hardware. This is designed to deal capably with the characteristics and requirements of mobile applications. This is the operating system for the appliance of mobile devices. In other words, you can say it the heart of the mobile systems. This is an essential component that operates the mobile device. FOR E.G. ANDROID, BLACKBERRY, IOS ,WINDOWS OS FOR MOBILES ETC.

Mobile Devices

Following is the list of most common forms of devices used in mobile computing:

1. Portable Computers

A portable computer is a computer that is designed in a way that you can move it from one place to another. It includes a display and a keyboard. Generally, portable computers are microcomputers.

Compaq Portable and Contemporary portable computer with 3 LCD screens were the early examples of portable computers. Now, portable computers are discontinued.

2. Personal Digital Assistant/Enterprise Digital Assistant (PDA or EDA)

A Personal Digital Assistant (PDA) is also known as a palmtop computer. Sometimes, it is also called Enterprise Digital Assistant (EDA). A personal Digital Assistant (PDA) is a mobile device used to function as a personal information manager or a personal data assistant. Its name, Personal Digital Assistant (PDA), was evolved from Personal Desktop Assistant, a software term for an application that prompts or prods the user of a computer with suggestions or provides a quick reference to contacts and other lists.

Apple Newton and UPOP PDA were the early examples of Personal Digital Assistant. Now, a Personal Digital Assistant (PDAs) are also discontinued.

3. Ultra-Mobile PC

An ultra-mobile PC was a small form factor version of a pen computer. It was a class of laptops whose specifications were launched by Microsoft and Intel in 2006.

Samsung q1 ultra-premium was the early example of an ultra-mobile PC. Now, ultra-mobile PCs are also discontinued.

4. Laptop

A laptop is a small, portable personal computer (PC) built in a foldable device. The folding structure of a laptop is called a clamshell form factor. The flip or clamshell is a form factor of a mobile phone or other devices that include two or more folded sections via a hinge. A laptop typically has a thin LCD or LED computer screen mounted on the inside of the clamshell's upper lid and an alphanumeric keyboard on the inside of the lower lid. Laptops are easy to carry for transportation, and that's why they are best suitable for mobile use.

5. Smartphone

A smartphone is a mobile device that combines cellular and mobile computing functions into one unit. The smartphones are invented to provide more advanced computing capability and connectivity than basic feature phones.

Smartphones are different from basic feature phones by their more robust hardware capabilities and extensive mobile operating systems, which facilitate more comprehensive software, internet i.e., web browsing over mobile broadband, and multimedia functionality i.e., music, video, cameras, and gaming etc., along with the core phone functions such as voice calls and text messaging.

6. Tablet Computers

A tablet computer is generally known as a tablet. It is a mobile computer with a mobile operating system and a touch-screen display processing circuit, and a rechargeable battery in a single, thin and flat unit. Tablets can do what other personal computers can do, but they don't have some input/output (I/O) abilities that computers have. Nowadays, tablets are very much similar to modern smartphones. The only difference is that tablets are relatively larger than smartphones, with screens 7 inches or larger and may not support a cellular network.

7. Wearable computers

Wearable computers are a type of computer that can be worn by the bearer under, with or on top of clothing. They are also known as body-borne computers or wearables, which are small electronic devices. Some examples of wearable computers are smartwatches, digital fitness bands etc.

Three-tier mobile computing architecture

First Tier/ Layer

• User Interface/Presentation Layer – deals with the user facing device handling & rendering.

• This tier includes a user interfacing components like Textbox, Labels, Checkboxes, etc.

Second Tier/Layer

• Process Management/application Layer – deals with Business logic & Rules.

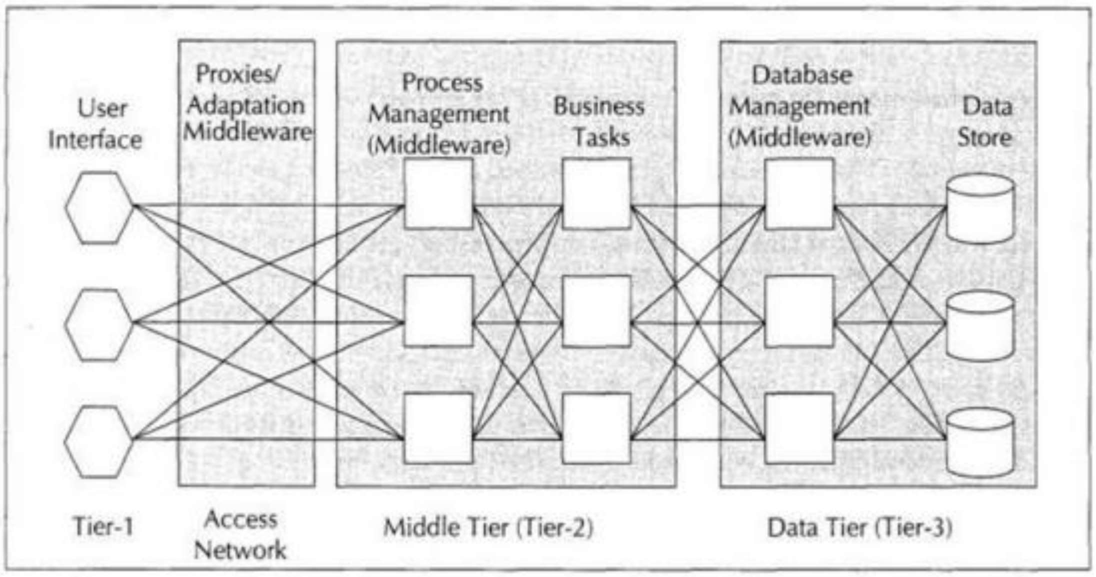
• It is capable of accommodating hundreds users.

• The middle process management tier controls transactions & asynchronous queuing to ensure reliable completion of transaction

Third Tier/Layer

• Database Management/Data Tier – deals with DB management & access.

• The three tier architecture is better suited for an effective networked client / server design



- These characteristics entertain the use of 3 tier architecture useful for internet applications & net centric systems

• To design a system for mobile computing , we need to keep in mind that the system will used through any network, any bearer, any agent, any device etc.

Android is a complete set of software for mobile devices such as tablet computers, notebooks, smartphones, electronic book readers, set-top boxes etc.

It contains a **linux-based Operating System**, **middleware** and **key mobile applications**.

It can be thought of as a mobile operating system. But it is not limited to mobile only. It is currently used in various devices such as mobiles, tablets, televisions etc.

It is developed by Google and later the OHA (Open Handset Alliance). Java language is mainly used to write the android code even though other languages can be used.

The goal of android project is to create a successful real-world product that improves the mobile experience for end users.

Features of Android

After learning what is android, let's see the features of android. The important features of android are given below:

1) It is open-source.

2) Anyone can customize the Android Platform.

3) There are a lot of mobile applications that can be chosen by the consumer.

4) It provides many interesting features like weather details, opening screen, live RSS (Really Simple Syndication) feeds etc.

It provides support for messaging services(SMS and MMS), web browser, storage (SQLite), connectivity (GSM, CDMA, Blue Tooth, Wi-Fi etc.), media, handset layout etc.

History of Android

The history and versions of android are interesting to know. The code names of android ranges from A to J currently, such as **Aestro**, **Blender**, **Cupcake**, **Donut**, **Eclair**, **Froyo**, **Gingerbread**, **Honeycomb**, **Ice Cream Sandwitch**, **Jelly Bean**, **KitKat** and **Lollipop**. Let's understand the android history in a sequence.

1) Initially, **Andy Rubin** founded Android Incorporation in Palo Alto, California, United States in October, 2003.

2) In 17th August 2005, Google acquired android Incorporation. Since then, it is in the subsidiary of Google Incorporation.

3) The key employees of Android Incorporation are **Andy Rubin**, **Rich Miner**, **Chris White** and **Nick Sears**.

4) Android is the nick name of Andy Rubin given by coworkers because of his love to robots.

5) In 2007, Google announces the development of android OS.

6) In 2008, HTC launched the first android mobile.

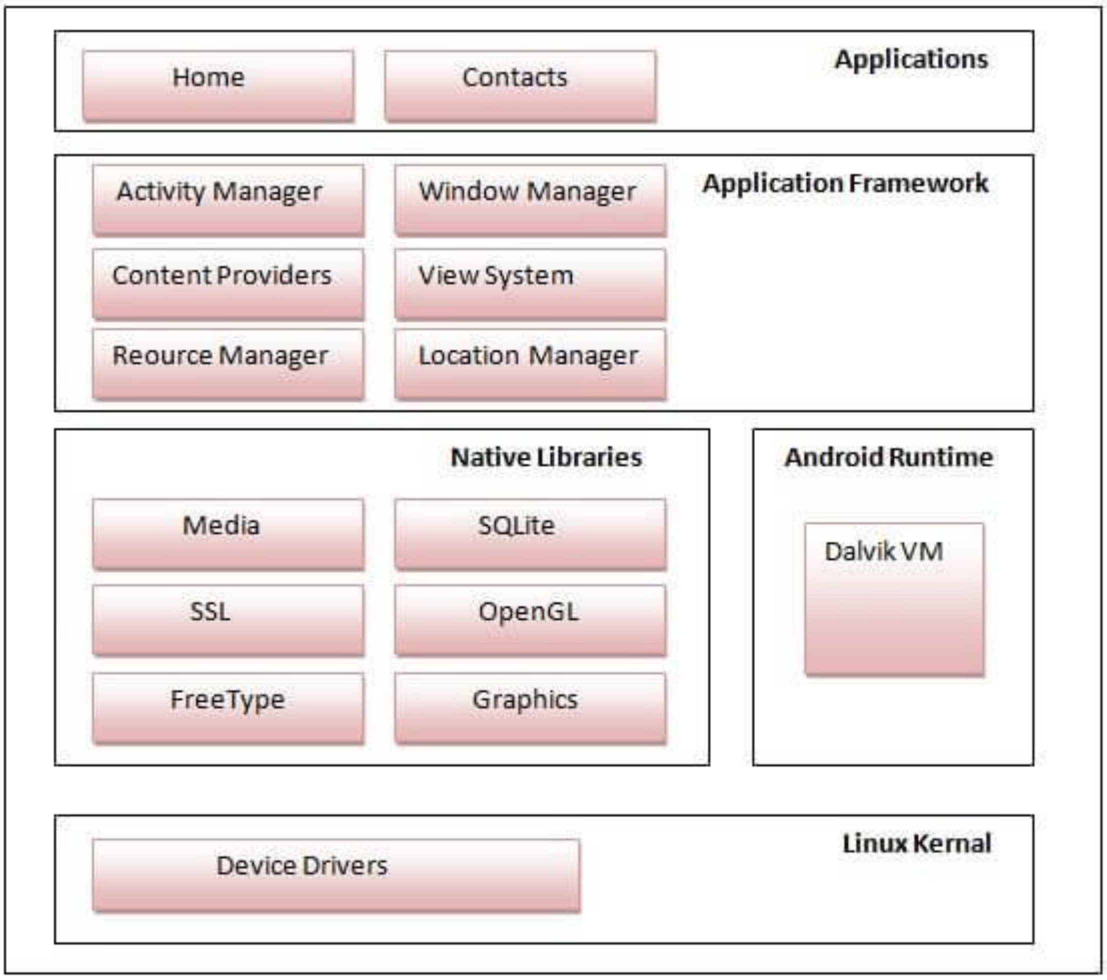
Version	**Code name**	**API Level**
1.5	Cupcake	3
1.6	Donut	4
2.1	Eclair	7
2.2	Froyo	8
2.3	Gingerbread	9 and 10
3.1 and 3.3	Honeycomb	12 and 13
4.0	Ice Cream Sandwitch	15
4.1, 4.2 and 4.3	Jelly Bean	16, 17 and 18
4.4	KitKat	19
5.0	Lollipop	21
6.0	Marshmallow	23
7.0	Nougat	24-25
8.0	Oreo	26-27

Android Architecture

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

1. linux kernel
2. native libraries (middleware),
3. Android Runtime
4. Application Framework
5. Applications

Let's see the android architecture first.



1) Linux kernel

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

---

2) Native Libraries

On the top of linux kernel, their are **Native libraries** such as WebKit, OpenGL, FreeType, SQLite, Media, C runtime library (libc) etc.

The WebKit library is responsible for browser support, SQLite is for database, FreeType for font support, Media for playing and recording audio and video formats.

---

3) Android Runtime

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

4) Android Framework

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

---

5) Applications

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

Dalvik Virtual Machine | DVM

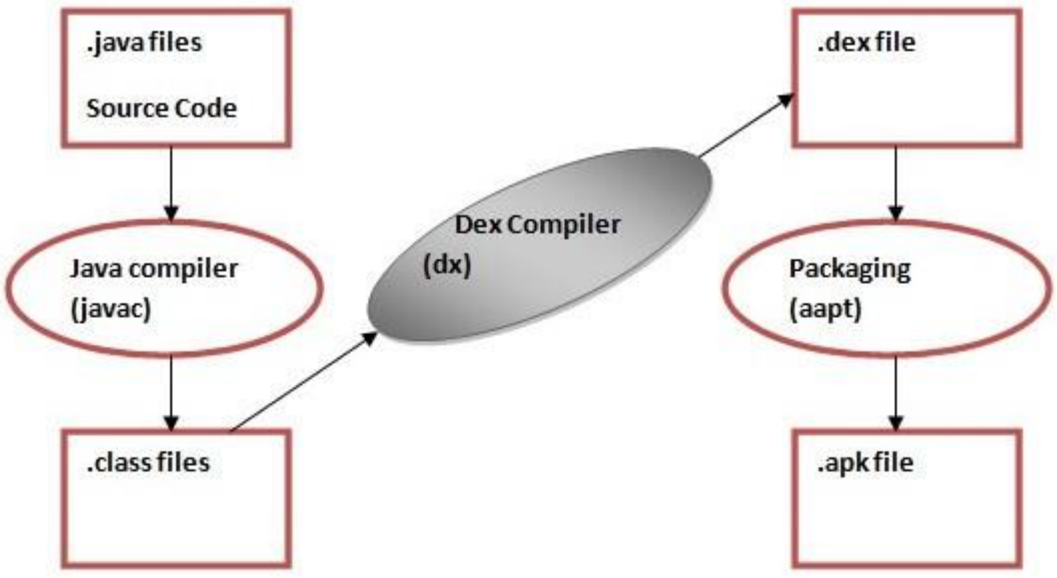
As we know the modern JVM is high performance and provides excellent memory management. But it needs to be optimized for low-powered handheld devices as well.

The **Dalvik Virtual Machine (DVM)** is an android virtual machine optimized for mobile devices. It optimizes the virtual machine for *memory*, *battery life* and *performance*.

Dalvik is a name of a town in Iceland. The Dalvik VM was written by Dan Bornstein.

The Dex compiler converts the class files into the .dex file that run on the Dalvik VM. Multiple class files are converted into one dex file.

Let's see the compiling and packaging process from the source file:



The **javac tool** compiles the java source file into the class file.

The **dx tool** takes all the class files of your application and generates a single .dex file. It is a platform-specific tool.

Android SDK and it's Components

Android SDK is a collection of libraries and Software Development tools that are essential for Developing Android Applications. Whenever Google releases a new version or update of Android Software, a corresponding SDK also releases with it. In the updated or new version of SDK, some more features are included which are not present in the previous version. Android SDK consists of some tools which are very essential for the development of Android Application. These tools provide a smooth flow of the development process from developing and debugging. Android SDK is compatible with all operating systems such as Windows, Linux, macOS, etc.

1. Android SDK Tools

Android SDK tool is an important component of Android SDK. It consists of a complete set of development and debugging tools. Below are the SDK developer tools:

- Android SDK Build tool.
- Android Emulator.
- Android SDK Platform-tools.
- Android SDK Tools.

Android SDK Build-Tools

Android SDK build tools are used for building actual binaries of Android App. The main functions of Android SDK Build tools are built, debug, run and test Android applications. The latest version of the Android SDK Build tool is 30.0.3. While downloading or updating Android in our System, one must ensure that its latest version is download in SDK Components.

3. Android Emulator

An Android Emulator is a device that simulates an Android device on your system. Suppose we want to run our android application that we code. One option is that we will run this on our Android Mobile by Enabling USB Debugging on our mobile. Another option is using Android Emulator. In Android Emulator the virtual android device is shown on our system on which we run the Android application that we code.

Thus, it simply means that without needing any physical device Android SDK component "Android Emulator" provides a virtual device on the System where we run our Application. The emulator's come with the configuration for Various android phones, tablets, Wear OS, and Android TV devices.

4. Android SDK Platform-tools

Android SDK Platform-tools is helpful when we are working on Project and they will show the error messages at the same time. It is specifically used for testing. It includes:

- Android Debug Bridge (ADB), is a command-line tool that helps to communicate with the device. It allows us to perform an action such as Installing App and Debugging App etc.
- Fastboot allows you to flash a device with a new system image.
- Systrace tools help to collect and inspect timing information. It is very crucial for App Debugging.

5. Android SDK Tools

Android SDK tool is a component of SDK tool. It consists of a set of tools which and other Utilities which are crucial for the development of Android Application. It contains the complete set of Debugging and Development tools for android.

Create and manage virtual devices

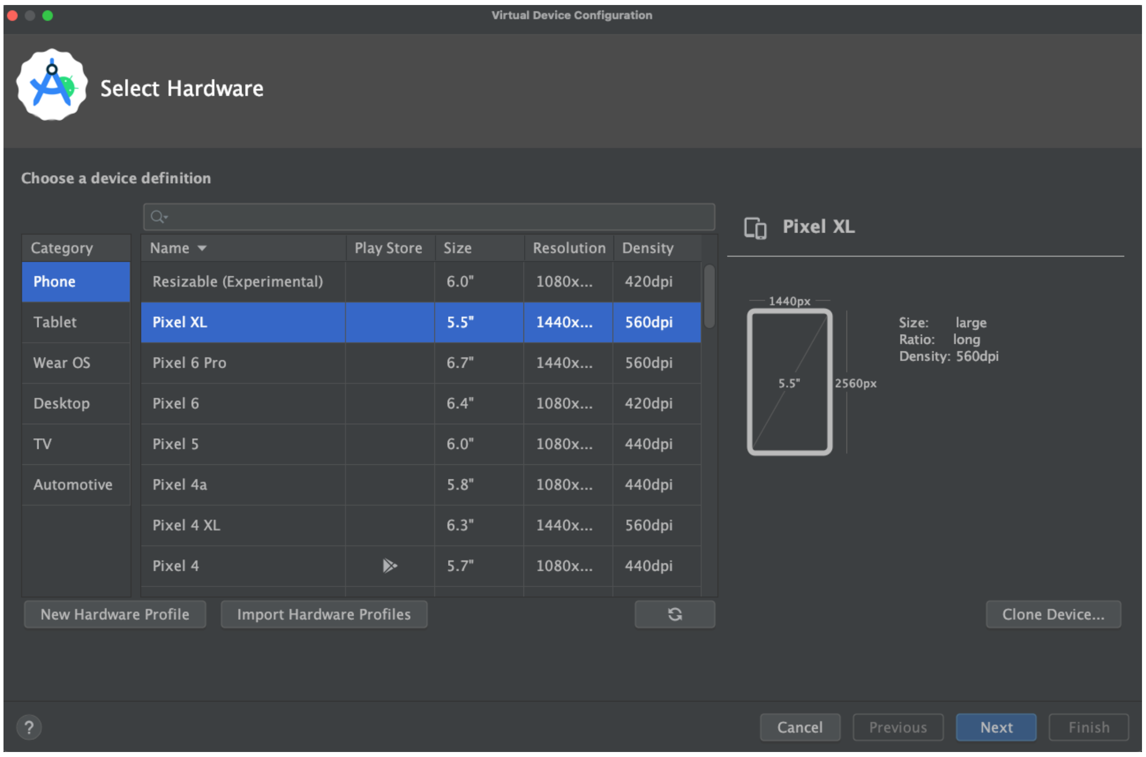
An Android Virtual Device (AVD) is a configuration that defines the characteristics of an Android phone, tablet, Wear OS, Android TV, or Automotive OS device that you want to simulate in the Android Emulator. The Device Manager is a tool you can launch from Android Studio that helps you create and manage AVDs.

Create an AVD

To create a new AVD:

1. Open the Device Manager.
2. Click **Create Device**.

The **Select Hardware** window appears.



Choose a device definition

Category	Name	Play Store	Size	Resolution	Density
Phone	Resizable (Experimental)		6.0"	1080x...	420dpi
Tablet	Pixel XL		5.5"	1440x...	560dpi
Wear OS	Pixel 6 Pro		6.7"	1440x...	560dpi
Desktop	Pixel 6		6.4"	1080x...	420dpi
TV	Pixel 5		6.0"	1080x...	440dpi
Automotive	Pixel 4a		5.8"	1080x...	440dpi
	Pixel 4 XL		6.3"	1440x...	560dpi
	Pixel 4	▶	5.7"	1080x...	440dpi

Pixel XL

1440px 5.5" 2560px

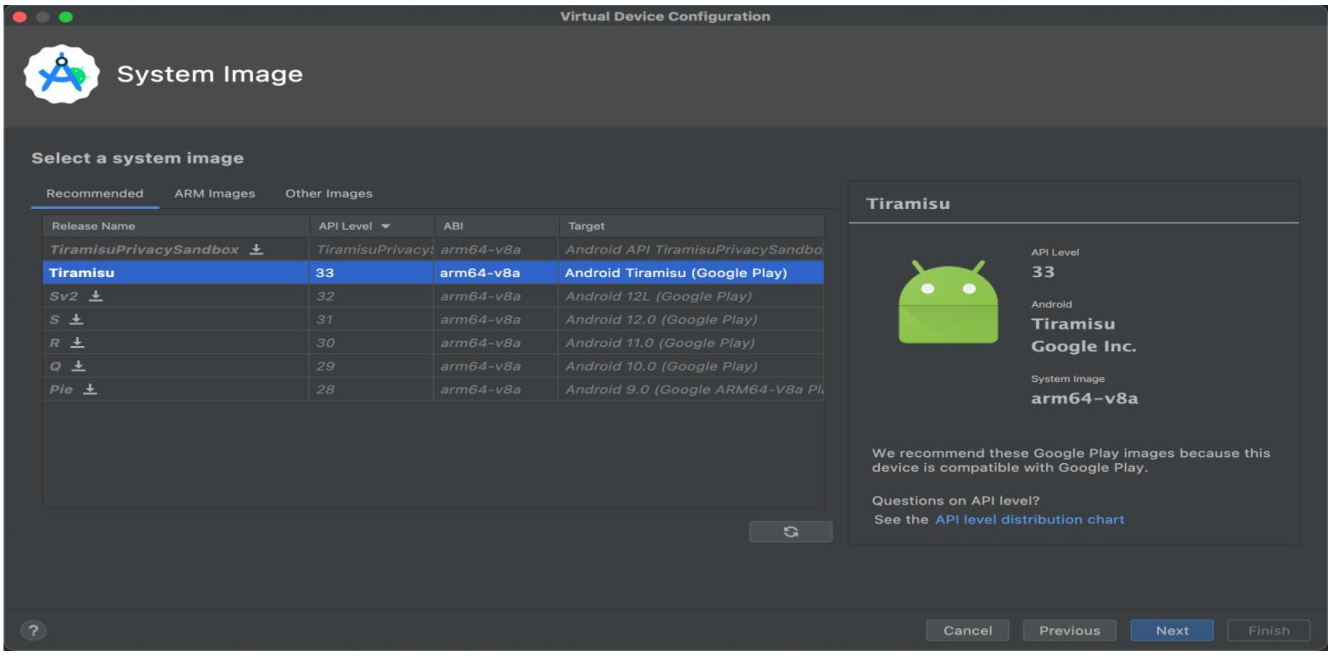
Size: large
Ratio: long
Density: 560dpi

Notice that only some hardware profiles include **Play Store**. These profiles are fully CTS compliant and might use system images that include the Play Store app.

3. Select a hardware profile, then click **Next**.

   If you don't see the hardware profile you want, you can create or import a hardware profile, as described in other sections on this page.

   The **System Image** window appears.



Select a system image

Release Name	API Level	ABI	Target
TiramisuPrivacySandbox	TiramisuPrivacy:	arm64-v8a	Android API TiramisuPrivacySandbo
Tiramisu	33	arm64-v8a	Android Tiramisu (Google Play)
Sv2	32	arm64-v8a	Android 12L (Google Play)
S	31	arm64-v8a	Android 12.0 (Google Play)
R	30	arm64-v8a	Android 11.0 (Google Play)
Q	29	arm64-v8a	Android 10.0 (Google Play)
Pie	28	arm64-v8a	Android 9.0 (Google ARM64-V8a Pl

Tiramisu

API Level
33

Android
Tiramisu
Google Inc.

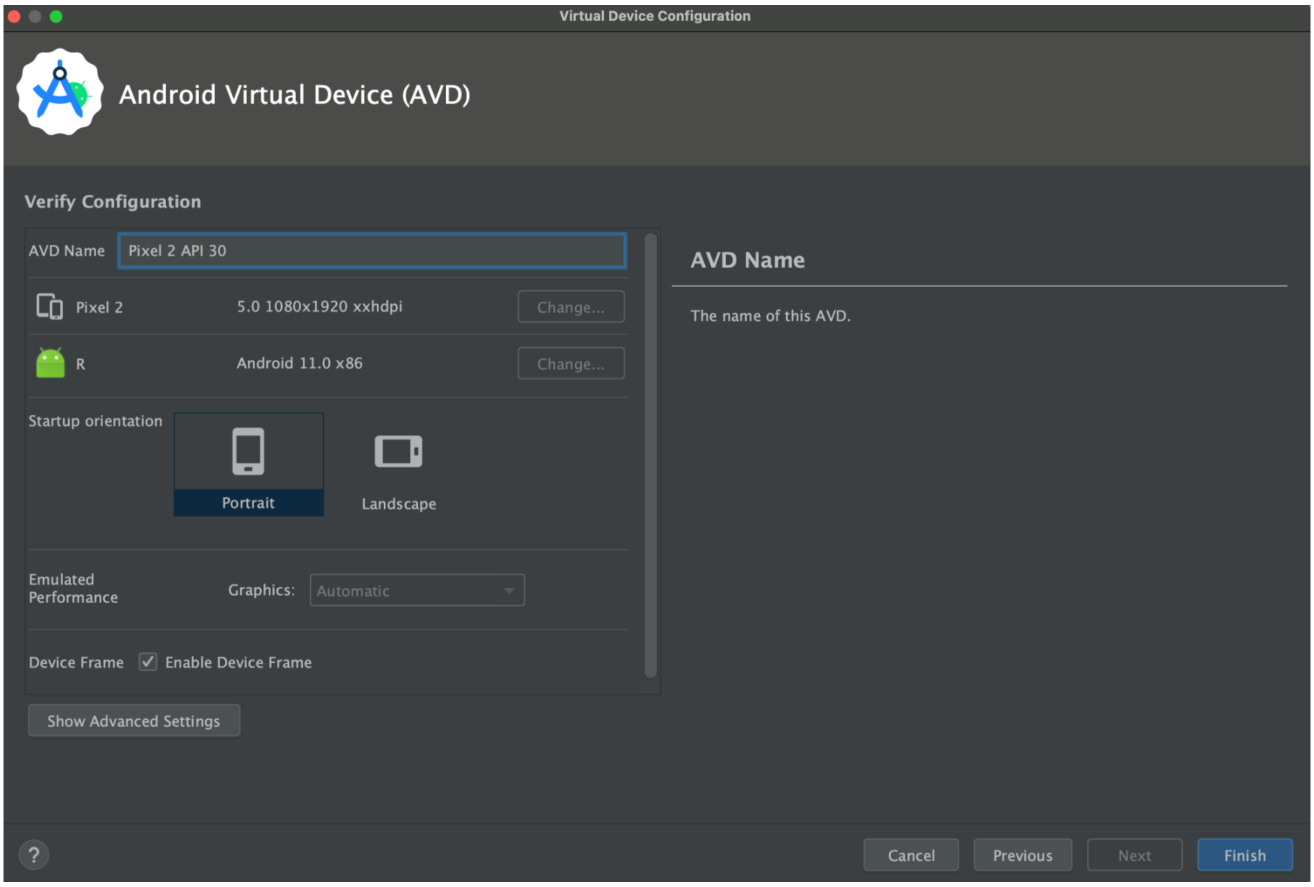
System Image
arm64-v8a

We recommend these Google Play images because this device is compatible with Google Play.

Questions on API level?
See the API level distribution chart

4. Select the system image for a particular API level, and then click **Next**.

   The **Verify Configuration** window appears.



Virtual Device Configuration

Android Virtual Device (AVD)

Verify Configuration

AVD Name: Pixel 2 API 30

Pixel 2 — 5.0 1080x1920 xxhdpi — Change...

R — Android 11.0 x86 — Change...

Startup orientation: Portrait / Landscape

Emulated Performance — Graphics: Automatic

Device Frame — ✓ Enable Device Frame

Show Advanced Settings

AVD Name

The name of this AVD.

5. Change the AVD properties as needed, and then click **Finish**.

   Click **Show Advanced Settings** to show more settings, such as the skin.

   The new AVD appears in the **Virtual** tab of the Device Manager and the target device menu.

   To create an AVD starting with a copy:

1. From the **Virtual** tab of the Device Manager, click **Menu** ⋮ and select **Duplicate**.

   The **Verify Configuration** window appears.

2. Click **Previous** if you need to make changes on the System Image or Select Hardware windows.

3. Make any changes you need, and then click **Finish**.

   The AVD appears in the **Virtual** tab of the Device Manager.

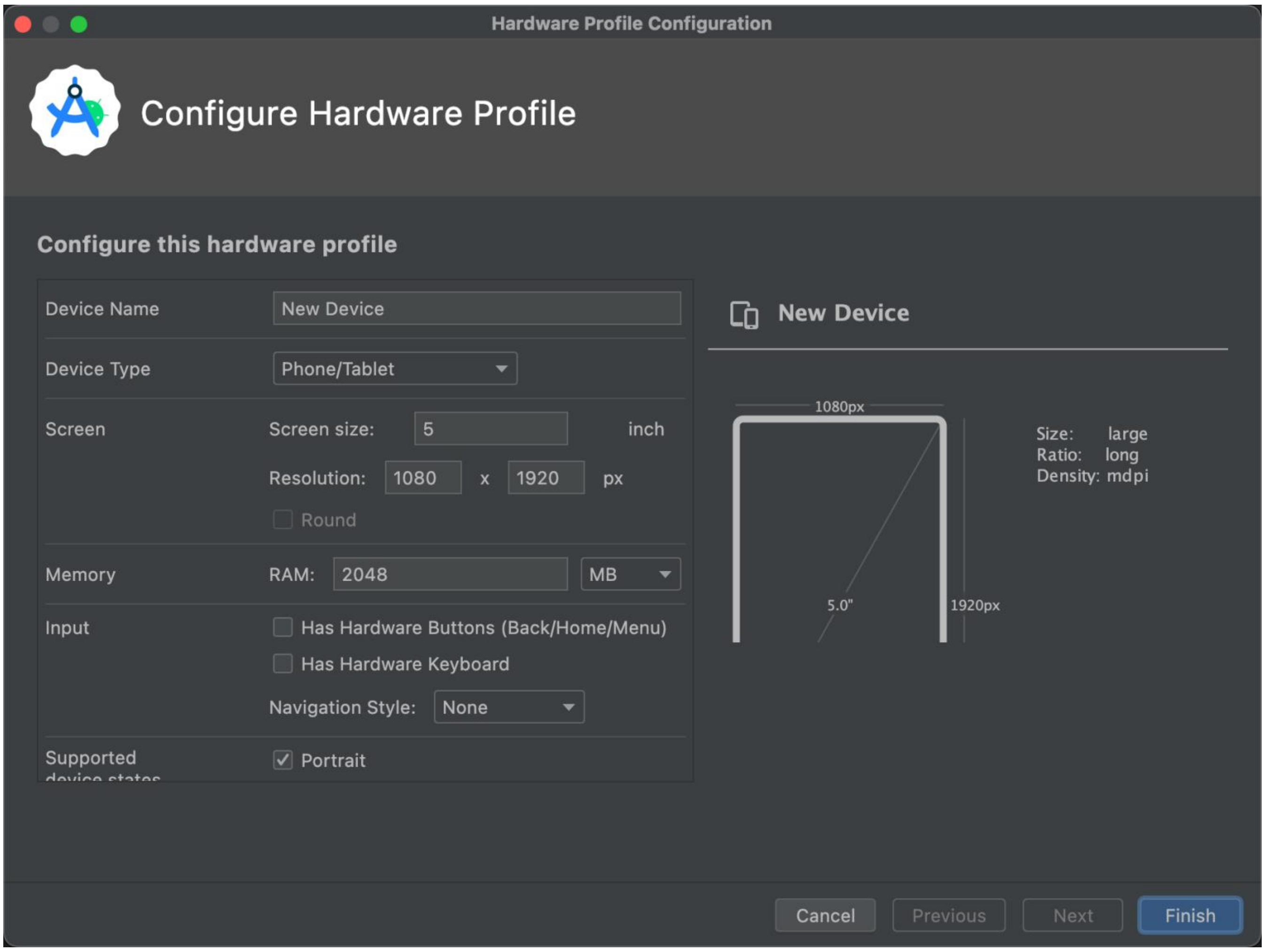
Create a hardware profile

The Device Manager provides predefined hardware profiles for common devices so you can easily add them to your AVD definitions. If you need to define a different device, you can create a new hardware profile.

You can define a new hardware profile from the beginning or copy a hardware profile as a starting point. The preloaded hardware profiles aren't editable.

To create a new hardware profile from the beginning:

1. In the Select Hardware window, click **New Hardware Profile**.
2. In the **Configure Hardware Profile** window, change the hardware profile properties as needed.



Hardware Profile Configuration

Configure Hardware Profile

Configure this hardware profile

Device Name	New Device					
Device Type	Phone/Tablet					
Screen	Screen size:	5		inch		
	Resolution:	1080	x	1920	px	
	☐ Round					
Memory	RAM:	2048	MB			
Input	☐ Has Hardware Buttons (Back/Home/Menu)					
	☐ Has Hardware Keyboard					
	Navigation Style:	None				
Supported device states	✓ Portrait					

New Device

Size: large
Ratio: long
Density: mdpi

3. Click **Finish**.

   Your new hardware profile appears in the **Select Hardware** window. You can create an AVD that uses the hardware profile by clicking **Next** or click **Cancel** to return to the **Virtual** tab or target device menu.

To create a hardware profile using a copy as a starting point:

1. In the **Select Hardware** window, select a hardware profile and click **Clone Device** or right-click a hardware profile and select **Clone**.
2. In the **Configure Hardware Profile** window, change the hardware profile properties as needed.
3. Click **Finish**.

   Your new hardware profile appears in the **Select Hardware** window. You can create an AVD that uses the hardware profile by clicking **Next** or click **Cancel** to return to the **Virtual** tab or target device menu.

Edit existing AVDs

You can perform the following operations on an AVD from the Device Manager's **Virtual** tab:

- To edit an AVD, click **Edit this AVD** and make your changes.
- To delete an AVD, click **Menu** and select **Delete**.
- To show the associated AVD INI and IMG files on disk, click **Menu** and select **Show on Disk**.
- To view AVD configuration details that you can include in bug reports to the Android Studio team, click **Menu** and select **View Details**.

Activity & Intents :-

An Activity represents a single screen in your app with which your user can perform a single, focused task such as taking a photo, sending an email, or viewing a map. An activity is usually presented to the user as a full-screen window.

An app usually consists of multiple screens that are loosely bound to each other. Each screen is an activity. Typically, one activity in an app is specified as the "main" activity (MainActivity.java), which is presented to the user when the app is launched. The main activity can then start other activities to perform different actions.

Each time a new activity starts, the previous activity is stopped, but the system preserves the activity in a stack (the "back stack"). When a new activity starts, that new activity is pushed onto the back stack and takes user focus. The back stack follows basic "last in, first out" stack logic. When the user is done with the current activity and presses the Back button, that activity is popped from the stack and destroyed, and the previous activity resumes.

An activity is started or activated with an *intent*. An Intent is an asynchronous message that you can use in your activity to request an action from another activity, or from some other app component. You use an intent to start one activity from another activity, and to pass data between activities.

An Intent can be *explicit* or *implicit*:

- An *explicit intent* is one in which you know the target of that intent. That is, you already know the fully qualified class name of that specific activity.
- An *implicit intent* is one in which you do not have the name of the target component, but you have a general action to perform.

User Interface:

Difference Between View and ViewGroup in Android

In Android Layout is used to describe the user interface for an app or activity, and it stores the UI elements that will be visible to the user. An android app's user interface is made up of a series of **View** and **ViewGroup** elements. In most cases, android apps will have one or more operations, each of which is a single screen of the app. Multiple UI components will be present in the operations, and those UI components will be instances of the View and ViewGroup subclasses. Generally, the android apps will contain one or more activities and each activity is one screen of the app. The activities will contain multiple UI components and those UI components are the instances of View and ViewGroup subclasses. In Android apps, the two very central classes are the **Android View class and ViewGroup class**. One or more tasks can be found in an Android app. A screen in an Android operation is identical to the windows in a desktop application. GUI components may be used in an operation. View or ViewGroup subclasses are used to build the GUI elements. **View** is a basic building block of UI (User Interface) in android. A view is a small rectangular box that responds to user inputs. Eg: EditText, Button, CheckBox, etc. **ViewGroup** is an invisible container of other views (child views) and other **ViewGroup.** Eg: **LinearLayout** is a **ViewGroup** that can contain other views in it. **ViewGroup** is a special kind of view that is extended from View as its base class. ViewGroup is the base class for layouts. As the name states View is singular and the group of Views is the **ViewGroup.** In simple terms, a view is a user interface feature that we interact with when we use an app, such as a button, editing text and images, and so on. **Android.view** has a child class called View. Observe While the View group is the container that houses all of these views as well as many other **ViewGroup** such as linear or Frame Layout. For example, if we design and use the **LinearLayout** as the root feature, our main layout would be the LinearLayout. Within it, we can add another view category (i.e. another LinearLayout) and several other views such as buttons or **TextViews**.

View

The View class is the base class or we can say that it is the superclass for all the GUI components in android. For example, the EditText class is used to accept the input from users in android apps, which is a subclass of View, and another example of the TextView class which is used to display text labels in Android apps is also a subclass of View.

Or the other definition,

View refer to the **android.view.View** class, which is the base class of all UI classes. android.view.View class is the root of the UI class hierarchy. So from an object point of view, all UI objects are View objects. Following are some of the common View subclasses that will be used in android applications.

- TextView
- EditText
- ImageView
- RadioButton
- Button
- ImageButton
- CheckBox
- DatePicker
- Spinner
- ProgressBar and etc.

These are some of the view subclass available in android.

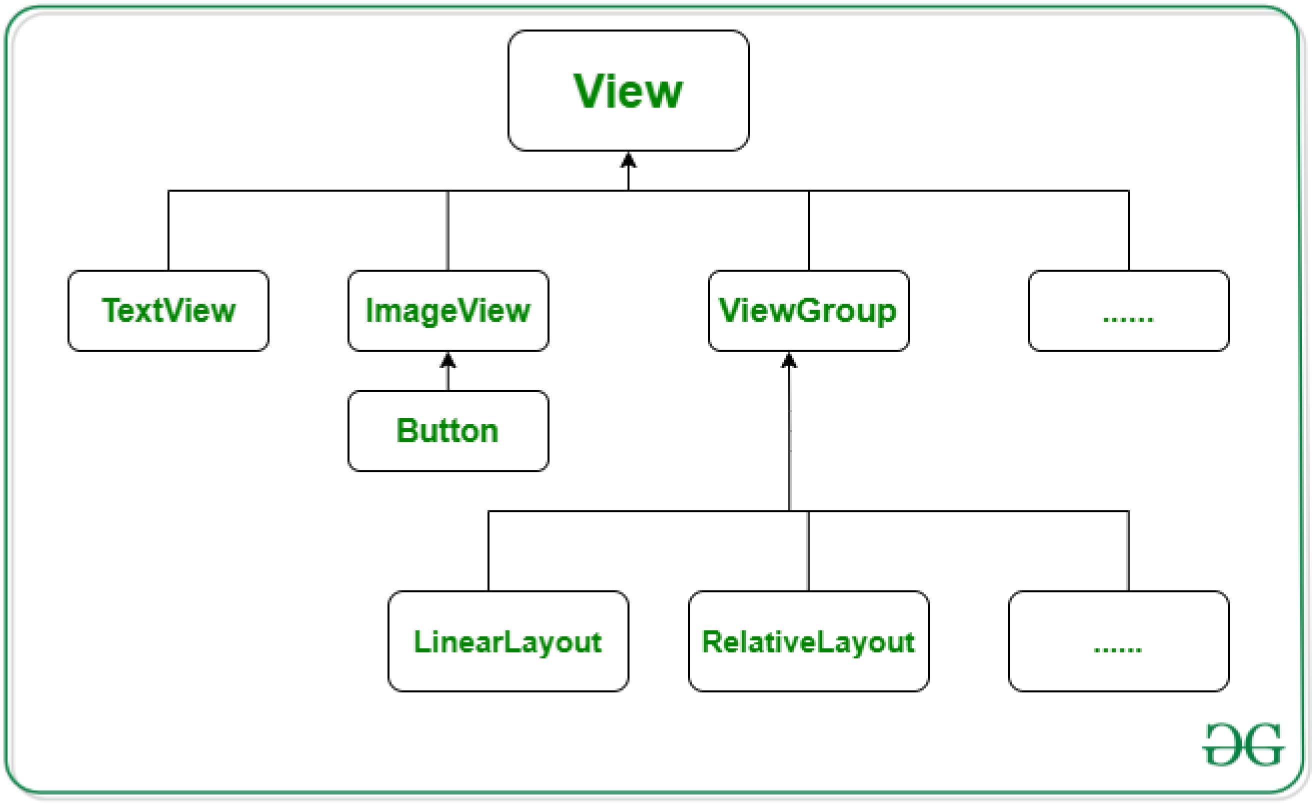
ViewGroup

The ViewGroup class is a subclass of the View class. And also it will act as a base class for layouts and layouts parameters. The ViewGroup will provide an invisible container to hold other Views or ViewGroups and to define the layout properties. For example, Linear Layout is the ViewGroup that contains UI controls like Button, TextView, etc., and other layouts also. **ViewGroup** Refer to the **android.view.ViewGroup** class, which is the base class of some special UI classes that can contain other View objects as children. Since ViewGroup objects are also View objects, multiple ViewGroup objects and View objects can be organized into an object tree to build a complex UI structure. Following are the commonly used ViewGroup subclasses used in android applications.

- FrameLayout
- WebView

- ListView
- GridView
- LinearLayout
- RelativeLayout
- TableLayout and many more.

The ViewGroup subclasses listed above group View instances together and takes care of their layout. For instance, the LinearLayout will render the components after each other either horizontally or vertically.



Difference Table

View	ViewGroup
View is a simple rectangle box that responds to the user's actions.	ViewGroup is the invisible container. It holds View and ViewGroup
View is the SuperClass of All component like TextView,	ViewGroup is a collection of Views(TextView, EditText, ListView,

View	**ViewGroup**
EditText, ListView, etc	etc..), somewhat like a container.
A View object is a component of the user interface (UI) like a button or a text box, and it’s also called a widget.	A ViewGroup object is a layout, that is, a container of other ViewGroup objects (layouts) and View objects (widgets)
Examples are EditText, Button, CheckBox, etc.	For example, LinearLayout is the ViewGroup that contains Button(View), and other Layouts also.
View refers to the android.view.View class	ViewGroup refers to the android.view.ViewGroup class
android.view.View which is the base class of all UI classes.	ViewGroup is the base class for Layouts.

So, these all are the basic key difference between the View class and Viewgroup class in Android.

Android - Location Based Services

Android location APIs make it easy for you to build location-aware applications, without needing to focus on the details of the underlying location technology.

This becomes possible with the help of **Google Play services**, which facilitates adding location awareness to your app with automated location tracking, geofencing, and activity recognition.

The Location Object

The **Location** object represents a geographic location which can consist of a latitude, longitude, time stamp, and other information such as bearing, altitude and velocity. There are following important methods which you can use with Location object to get location specific information −

Sr.No.	Method & Description
1	**float distanceTo(Location dest)** Returns the approximate distance in meters between this location and the given location.
2	**float getAccuracy()** Get the estimated accuracy of this location, in meters.
3	**double getAltitude()** Get the altitude if available, in meters above sea level.
4	**float getBearing()** Get the bearing, in degrees.
5	**double getLatitude()** Get the latitude, in degrees.
6	**double getLongitude()** Get the longitude, in degrees.
7	**float getSpeed()** Get the speed if it is available, in meters/second over ground.
8	**boolean hasAccuracy()** True if this location has an accuracy.
9	**boolean hasAltitude()** True if this location has an altitude.
10	**boolean hasBearing()** True if this location has a bearing.
11	**boolean hasSpeed()** True if this location has a speed.
12	**void reset()** Clears the contents of the location.
13	**void setAccuracy(float accuracy)** Set the estimated accuracy of this location, meters.
14	**void setAltitude(double altitude)**

	Set the altitude, in meters above sea level.
15	**void setBearing(float bearing)** Set the bearing, in degrees.
16	**void setLatitude(double latitude)** Set the latitude, in degrees.
17	**void setLongitude(double longitude)** Set the longitude, in degrees.
18	**void setSpeed(float speed)** Set the speed, in meters/second over ground.
19	**String toString()** Returns a string containing a concise, human-readable description of this object.

Get the Current Location

To get the current location, create a location client which is **LocationClient** object, connect it to Location Services using **connect()** method, and then call its **getLastLocation()** method. This method returns the most recent location in the form of **Location** object that contains latitude and longitude coordinates and other information as explained above. To have location based functionality in your activity, you will have to implement two interfaces −

- GooglePlayServicesClient.ConnectionCallbacks
- GooglePlayServicesClient.OnConnectionFailedListener

Android SQLite

SQLite is an **open-source relational database** i.e. used to perform database operations on android devices such as storing, manipulating or retrieving persistent data from the database.

It is embedded in android bydefault. So, there is no need to perform any database setup or administration task.

Here, we are going to see the example of sqlite to store and fetch the data. Data is displayed in the logcat. For displaying data on the spinner or listview, move to the next page.

SQLiteOpenHelper class provides the functionality to use the SQLite database.

Backward Skip 10sPlay VideoForward Skip 10s

SQLiteOpenHelper class

The android.database.sqlite.SQLiteOpenHelper class is used for database creation and version management. For performing any database operation, you have to provide the implementation of **onCreate()** and **onUpgrade()** methods of SQLiteOpenHelper class.

Constructors of SQLiteOpenHelper class

There are two constructors of SQLiteOpenHelper class.

Constructor	**Description**
SQLiteOpenHelper(Context context, String name, SQLiteDatabase.CursorFactory factory, int version)	creates an object for creating, opening and managing the database.
SQLiteOpenHelper(Context context, String name, SQLiteDatabase.CursorFactory factory, int version, DatabaseErrorHandler errorHandler)	creates an object for creating, opening and managing the database. It specifies the error handler.

Methods of SQLiteOpenHelper class

There are many methods in SQLiteOpenHelper class. Some of them are as follows:

Method	**Description**
public abstract void onCreate(SQLiteDatabase db)	called only once when database is created for the first time.

public abstract void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)	called when database needs to be upgraded.
public synchronized void close ()	closes the database object.
public void onDowngrade(SQLiteDatabase db, int oldVersion, int newVersion)	called when database needs to be downgraded.

SQLiteDatabase class

It contains methods to be performed on sqlite database such as create, update, delete, select etc.

Methods of SQLiteDatabase class

There are many methods in SQLiteDatabase class. Some of them are as follows:

Method	**Description**
void execSQL(String sql)	executes the sql query not select query.
long insert(String table, String nullColumnHack, ContentValues values)	inserts a record on the database. The table specifies the table name, nullColumnHack doesn't allow completely null values. If second argument is null, android will store null values if values are empty. The third argument specifies the values to be stored.
int update(String table, ContentValues values, String whereClause, String[] whereArgs)	updates a row.
Cursor query(String table, String[] columns, String selection, String[] selectionArgs, String groupBy, String having,	returns a cursor over the resultset.

String orderBy)	