Date - 6.2.2021

**REVISION NOTES**

NETWORKING IN ANDROID

* Networking – Sending and receiving data from mobile app to Internet or any other network computer, is called Networking. Data includes messages, photoes, videos etc.
* An app in android with networking capabilities is called network application.
* Sending network request using Library is quick and easy way to work with network.
* In out course we use volley library to do the same. Google developed volley library for networking.
* To Do networking we need to set below permissions in android manifest file otherwise we will get a run time errors.
* android.permission.INTERNET permission.

To access the Internet from application

* android.permission.ACCESS\_NETWORK\_STATE permission

To check the network state in your app

* Data can be download or uploaded from and to server or any other computer using any one the two method available in apache HttpClient Library.
* HttpGet

Most used to download data using get method of http protocol, however data can be send in key value pair.

* HttpPost

Mostly used to post data in any form like text, image, video etc.

Can also used download data.

> Difference between GET method & POST method :

|  |  |
| --- | --- |
| **GET** | **POST** |
| In GET method, values are visible in the URL. | In POST method, values are not visible in the URL. |
| GET has a limitation on the length of the values, generally 255 characters. | POST has no limitation on the length of the values since they are submitted via the body of HTTP. |
| GET is less secure compared to POST because data sent is part of the URL | POST is a little safer than GET because the parameters are not stored in browser history or in web server logs |
| GET performs are better compared to POST because of the simple nature of appending the values in the URL. | It has lower performance as compared to GET method because of time spent in including POST values in the HTTP body. |
| This method supports only string data types. | This method supports different data types, such as string, numeric, binary, etc. |
| GET results can be bookmarked. | POST results cannot be bookmarked. |
| GET request is often cacheable. | The POST request is hardly cacheable. |
| GET Parameters remain in web browser history. | Parameters are not saved in web browser history. |

* Downloading data is an long running task and it is recommended that all the long running task should be performed not in UI thread otherwise we may get force close error in application.
* Generally we are using following libraries in Android:
* Material Library
* Butterknife library
* Volley Library
* Picasso Library
* Bannerslider Library
* What is Library in Android ?
* Android library projects provide reusable code and resources that can save your time and efforts when developing apps
* A library project is merged into an Android PacKage (APK) file (with file extension .apk) when an app project is built.
* Volley Library Integration Steps :

If we want to use networking in any project, then we have to perform following 4 steps:

1. Include volley version in dependencies of app level build.gradle file.
2. Make new class “LruBitmapCache” and copy the readymade code given by sir in it.
3. Make new class “AppController” and copy the readymade code given by sir in it.
4. Open AndroidManifest.xml and add mentioned code in <application/> node. Also add two permissions above <application/> node.

* PHP (Hypertext Preprocessor):

\* PHP is server side scripting language, so its code executes on server.

\* It is used in Web Development and can be embedded into HTML. It is also used to develop web services (part of web development).

> What PHP can do the following :

- Generate dynamic page content

- Create, open, read, write and close the files on the server.

- Collect form data.

- Send and receive cookies(data which transfers between client & server).

- Add, delete & modify data in database.

- Encrypt or decrypt data.

> What is WAMP ?

- It is a package of independently created programs installed on computers that uses microsoft windows operating system.

- WAMP is an acronym of Wndows, Apache, MySQL, and one of (PHP, Perl or Python).

- Apache is web server, MySQL is database, PHP Perl & Python are scripting languages.

- Other programs may also be included in a package, such as phpMyAdmin which provides a graphical user interface for the MySQL database manager.

> In this project, we need to learn the following :

- PHP, HTML, Bootstrap, MySQL, phpMyAdmin

> What is Web-Services ?

- Web Service is a function (programatic interface) on server which can be called by any app with any language.

- It is used to send and receive data between different software applications and different plateforms.

- We can share business logic, data & processes through web service.

- Web service do not have any GUI.

> What is JSON ?

- JSON is a data interchange format used to communicate between application and server, between server to server or between application to application.

- JSON – JavaScript Object Notation.

- Basic JSON message (JSONObject) consists of key-value pairs, like

[{“name” : ”Gaurav”}, {“address” : ”Hilldrive”}]

- Here [ ] means array and { } means object.

- Output from web services in JSON format must be within [ ] array.

> How to generate JSON Object using PHP ?

- First create array in PHP.

- PHP has json\_encode() function for encoding array in JSON format. So pass the array in this function.

- echo the values, which above function returns.

- It must be used only once per page or end of any web page.

LAYOUTS IN ANDROID

> What is Layouts in Android ?

- Android Layouts is used to define the user interface which holds the UI controls or widgets that will appear on the screen of an android application or activity.

- View : View is anything that an Android Application can display. It can be any type of widget like image, text etc.

View is the baseclass for all UI components.

- ViewGroup : It is an invisible container of views and other viewgroups. Layouts can be called viewgroups.

ViewGroup is the base class for all layouts. ViewGroup extends View class.

- As pe google’s suggestion, don’t use layout within layout, if possible.

- Also use Constraint Layout instead of any other layout. We can reduce the use of layouts within layouts by using constraint layout.

> Types of Layouts in Android : (Click on link to see detail, Reference websites : www.technxt.net and [www.tutorialspoint.com](http://www.tutorialspoint.com)) :

* [**Linear Layout**](http://technxt.net/linear-layout-in-android-android-linear-layout-example/) :- LinearLayout is a view group used to provide child View elements one by one that aligns all children in a single direction, vertically or horizontally.
* Related main attributes are orientation, weightSum etc…
* [**Relative Layout**](http://technxt.net/relative-layout-in-android-relative-layout-example/):- RelativeLayout is a view group that displays child views in relative positions.
* [**Constraint Layout**](https://developer.android.com/reference/androidx/constraintlayout/widget/ConstraintLayout) :- A ConstraintLayout is a [android.view.ViewGroup](https://developer.android.com/reference/android/view/ViewGroup/package-summary) which allows you to position and size widgets in a flexible way. ConstraintLayout is a ViewGroup subclass, used to specify the position of a layout constraints for every child View relative to other views present. A ConstraintLayout is similar to a RelativeLayout, but having more power.
* [**Frame Layout**](https://www.tutorialspoint.com/android/android_frame_layout.htm) :- FrameLayout is a ViewGroup subclass which is used to specify the position of View instances it contains on the top of each other to display only a single View inside the FrameLayout.
* You can, however, add multiple children to a FrameLayout and control their position within the FrameLayout by assigning gravity to each child, using the android:layout\_gravity attribute.
* [**Absolute Layout**](https://www.tutorialspoint.com/android/android_absolute_layout.htm):- AbsoluteLayout enables you to specify the exact location of its children.
* [**Table Layout**](https://www.tutorialspoint.com/android/android_table_layout.htm):- TableLayout is a view that groups views into rows and columns. weightSum attribute is used to equally divide rows & columns in table.
* [**List View**](http://technxt.net/how-to-create-listview-in-android/) :- ListView is a view group that displays a list of scrollable items.
* [**Grid View**](http://technxt.net/gridview-in-android-android-gridview-example/) :- GridView is a ViewGroup that displays items in a two-dimensional, scrollable grid.

**CoordinatorLayout** :- A CoordinatorLayout (a ViewGroup) brings the different elements (child Views) of a layout into a harmonious or efficient relationship:

With the help of a CoordinatorLayout, child views work together harmoniously to implement awesome behaviours such as **drags, swipes, flings or any other gestures**.

<https://guides.codepath.com/android/handling-scrolls-with-coordinatorlayout>

> Note : In mipmap folder, we save images in png, jpg, bmp etc format.

In drawable folder, we save xml files of images, backgrounds etc.

**FRAGMENTS**

**Android Fragment** is the part of activity, it is also known as sub-activity. There can be more than one fragment in an activity. Fragments represent multiple screen inside one activity.

A [Fragment](https://developer.android.com/reference/androidx/fragment/app/Fragment) represents a reusable portion of your app's UI. A fragment defines and manages its own layout, has its own lifecycle, and can handle its own input events. Fragments cannot live on their own--they must be hosted by an activity or another fragment.

<https://li2.gitbooks.io/android-programming-journey/content/Activities/what-is-the-difference-between-a-fragment-and-an-activity.html>

<https://www.quora.com/How-do-you-differentiate-the-activity-and-fragment-in-Android>

<https://guides.codepath.com/android/creating-and-using-fragments>

> Advantages/Need of Fragments :

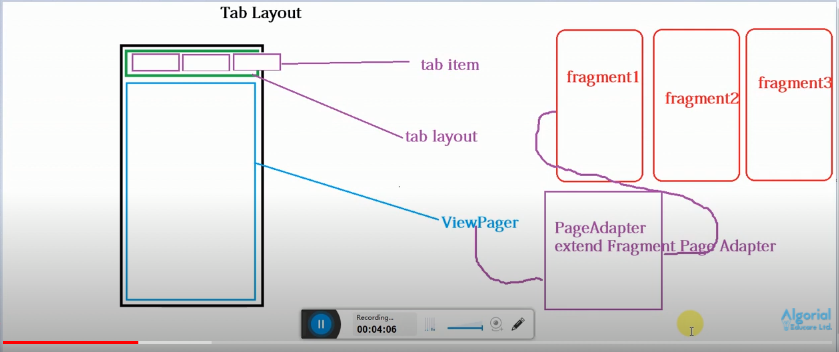
- Multiple activity uses separate multiple resources. If we use more fragments in one activity then it will use resources provided to that activity only, not separate resources.

So we can do several activity’s work in one activity. So less space consumed by use of fragments. So we can reduce out app size.

- Speed is also increase bcoz we don’t have to switch one activity to another frequenty.

- Reusability of code- fragments have their own layout, so it can be used across many different activities.

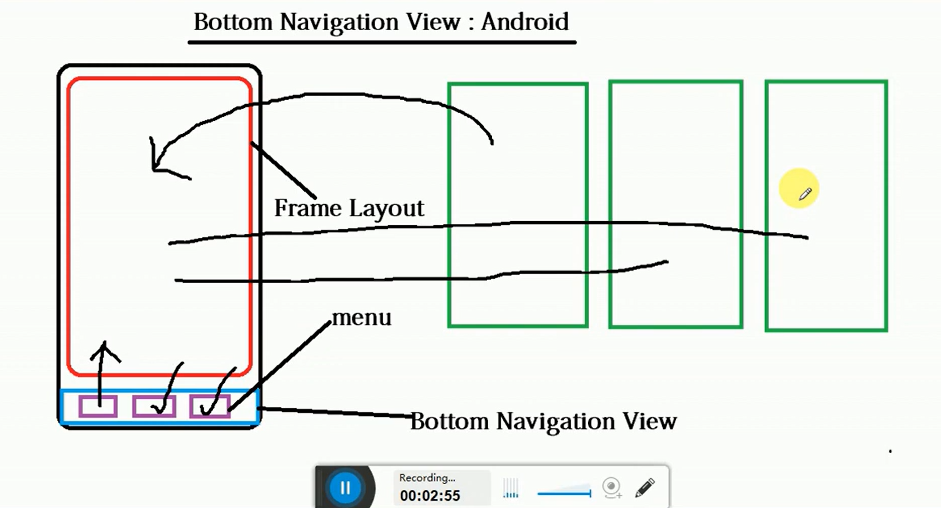
> Tab Layout with Fragments : (MD Jamal video 4)



- Tabs organize content across different screens, data sets, and other interactions.

- Tabs organize and allow navigation between groups of content that are related and at the same level of hierarchy.

> Bottom Navigation View using Fragments : (MD Jamal video 5)



> Defference between appbar, Actionbar & Toolbar :

<https://www.geeksforgeeks.org/difference-between-appbar-actionbar-and-toolbar-in-android/>

**SQLite Database**

**SQLite** is an **open-source relational database**. It is embedded in android bydefault.

What is RDBMS

**RDBMS** stands for *Relational Database Management Systems.*.

All modern database management systems like SQL, MS SQL Server, IBM DB2, ORACLE, My-SQL and Microsoft Access are based on RDBMS.

It is called Relational Data Base Management System (RDBMS) because it is based on relational model introduced by E.F. Codd.

## **How it works**

Data is represented in terms of tuples (rows) in RDBMS.

Relational database is most commonly used database. It contains number of tables and each table has its own primary key.

Due to a collection of organized set of tables, data can be accessed easily in RDBMS.

## **What is table**

The RDBMS database uses tables to store data. A table is a collection of related data entries and contains rows and columns to store data.

A table is the simplest example of data storage in RDBMS.

## **NULL Values**

The NULL value of the table specifies that the field has been left blank during record creation. It is totally different from the value filled with zero or a field that contains space.

Difference between DBMS and RDBMS

Although DBMS and RDBMS both are used to store information in physical database but there are some remarkable differences between them.

The main differences between DBMS and RDBMS are given below:

|  |  |
| --- | --- |
| **DBMS** | **RDBMS** |
| DBMS applications store **data as file**. | RDBMS applications store **data in a tabular form**. |
| In DBMS, data is generally stored in either a hierarchical form or a navigational form. | In RDBMS, the tables have an identifier called primary key and the data values are stored in the form of tables. |
| DBMS uses file system to store data, so there will be **no relation between the tables**. | in RDBMS, data values are stored in the form of tables, so a **relationship** between these data values will be stored in the form of a table as well. |
| DBMS has to provide some uniform methods to access the stored information. | RDBMS system supports a tabular structure of the data and a relationship between them to access the stored information. |
| DBMS **does not support distributed database**. | RDBMS **supports distributed database**. |
| DBMS is meant to be for small organization and **deal with small data**. it supports **single user**. | RDBMS is designed to **handle large amount of data**. it supports **multiple users**. |
| Examples of DBMS are file systems, **xml** etc. | Example of RDBMS are **mysql**, **postgre**, **sql server**, **oracle** etc. |

After observing the differences between DBMS and RDBMS, you can say that RDBMS is an extension of DBMS. There are many software products in the market today who are compatible for both DBMS and RDBMS. Means today a RDBMS application is DBMS application and vice-versa.

> Note : If we done any changes to table structure (eg. Change in column name etc), we have to uninstall the app from mobile and re-install it.

*// https://www.youtube.com/watch?v=-HYg8rXzX-I for CRUD operation video by ForMyScholars.*

> To create a table in SQLite database :

1) Write query in constant as per below:

**private static final** String ***CREATE\_COURSE*** = **"create table IF NOT EXISTS "** + ***COURSE*** +  
 **"(\_id integer PRIMARY KEY AUTOINCREMENT, name text, duration real, fees int, detail text)"**;  
  
**private static final** String ***CREATE\_INQUIRY*** = **"create table IF NOT EXISTS "** + ***INQUIRY*** +  
 **"(\_id integer PRIMARY KEY AUTOINCREMENT, fullname text, email text, mobile text, gender int, inquiry\_date text,learning\_mode int, qualification text, preferred\_time text, last\_followup\_detail text, coursetitle text)"**;

2) Then run query as per below:

**try**{  
 db.execSQL(***CREATE\_COURSE***); *// table will be created* db.execSQL(***CREATE\_INQUIRY***); *// table will be created*}  
**catch**(SQLiteException error)  
{  
 log.*e*(error.getMessage());  
 Toast.*makeText*(**ctx**, error.getMessage(), Toast.***LENGTH\_LONG***).show();  
}  
log.*d*(**"Table Created"**);

> Syntax of Queries as per first approach :

> First create SQLiteDatabase object :

SQLiteDatabase db = this.getWritableDatabase();

1) Insert:

**sql** = **"insert into"** + DBAdapter.***COURSE*** + **"(name, duration, fees, detail) values ('PHP', 6, 8500, 'some detail goes here')"**;

**db**.execSQL(sql);

2) Update:

**sql** = **"update"** + DBAdapter.***COURSE*** + **"set fees=9999, detail='Its Android app development course' where \_id=2"**;

**db**.execSQL(sql);

3) Delete:

**sql** = **"delete from"** + DBAdapter.***COURSE*** + **"where \_id=2"**;

**db**.execSQL(sql);

4) Select:

**sql** = **"select \* from "** + DBAdapter.***COURSE***;

Cursor c = **null**; *// Cursor variable c is used to hold the result of the query.*log.*d*(sql);  
**try**{  
 c = **db**.rawQuery(sql, **null**);  
}  
**catch**(SQLiteException error)  
{  
 log.*e*(error.getMessage());  
 Toast.*makeText*(**ctx**, error.getMessage(), Toast.***LENGTH\_LONG***).show();  
}  
**return** c;  
*// c can be null if there is error and not null if there is no error.*

> Syntax of Queries as per second approach :

> First create SQLiteDatabase object as per below :

**SQLiteDatabase db** = **this**.openOrCreateDatabase(DBAdapter.***DATABASE***, SQLiteDatabase.***OPEN\_READWRITE***,**null**);

1) Insert:

ContentValues values = **new** ContentValues();  
values.put(**"fullname"**, **fullname**);  
values.put(**"email"**,**email**);  
values.put(**"mobile"**,**mobile**);

long response = 0;

response = **db**.insert(DBAdapter.***INQUIRY***, **null**, values); *//(press ctr+q on insert query). Insert query returns long the row ID of the newly inserted row, or -1 if an error occurred*

2) Update:

String Conditions[] = {String.*valueOf*(**InquiryId**)};  
response = **db**.update(DBAdapter.***INQUIRY***, values, **"\_id=?"**, Conditions);

3) Delete:

String Conditions[] = {String.*valueOf*(CurrentInquiry.getId())};  
**db**.delete(DBAdapter.***INQUIRY***, **"\_id=?"**, Conditions);  
**InquiryList**.remove(position);  
notifyDataSetChanged();

4) Select:

String ColumnList[] = {**"\_id"**,**"fullname"**,**"mobile"**,**"qualification"**,**"coursetitle"**,**"inquiry\_date"**,**"learning\_mode"**,**"gender"**};  
Cursor c = **db**.query(DBAdapter.***INQUIRY***, ColumnList, **null**, **null**, **null**, **null**, **"\_id desc"**);

**SERVICES IN ANDROID**

> There are 3 main application components in Android :

1) Activity 2) Broadcast Receivers and 3) Services.

> Services :

Example - <https://www.geeksforgeeks.org/services-in-android-with-example/>

A Service is an application component that can perform long-running operations in the background and does not provide a user interface.

> A service can handle network transactions, play music, perform file I/O, or interact with a content provider, all from the background.

> MP3 Music players and sports-score monitors are examples of applications that should continue to run and update without an interactive visual component (Activity) visible.

 Following are the three different types of services:

1. Foreground

 A foreground service performs some operation that is noticeable to the user.

 For example, an audio app would use a foreground service to play an audio track.

 Foreground services must display a Notification.

 Foreground services continue running even when the user isn't interacting with the app.

1. Background

 A background service performs an operation that isn't directly noticed by the user. For example, if an app used a service to compact its storage, that would usually be a background service.

 Note: If your app targets API level 26(Android 8- Oreo) or higher, the system imposes restrictions on running background services when the app itself isn't in the foreground.

 In most situations, for example, you shouldn't access location information from the background. Instead, schedule tasks using Work Manager.

1. Bound

 A service is bound when an application component binds to it by calling bindService().

 A bound service offers a client-server interface that allows components to interact with the service, send requests, receive results, and even do so across processes with interprocess communication (IPC).

 A bound service runs only as long as another application component is bound to it.

 Multiple components can bind to the service at once, but when all of them unbind, the service is destroyed.

WILL IT BE STOPPED BY ANDROID?

 The Android system will force-stop a service only when memory is low and it must recover system resources for the activity that has user focus.

 If the service is bound to an activity that has user focus, then it's less likely to be killed, and if the service is declared to run in the foreground then it will almost never be killed.

 Otherwise, if the service was started and is longrunning, then the system will lower its position in the list of background tasks over time and the service will become highly susceptible to killing

> Coding in MyBackgroundService.java file :

> First extend Service class and implement its cumpulsary method onBind().

> Then from code menu, override the following methods;

1) onCreate()

2) onStartCommand()

Runnable BackgroundTask = **new** Runnable() {  
 @Override  
 **public void** run() {  
 PlayMantra(); // Do any task that you want to run in background.  
 }  
};  
BackgroundTask.run();  
**return super**.onStartCommand(intent, flags, startId);

1. onDestroy()

> Example of Background & Foreground Service :

- We are watching live streaming of any serial episode. So the data will be downloaded in the background for live streaming. This is background service.

- Another example, windows 10 automatically updates its software in background when internet is on. This is also Background service.

- This will not close when we minimuze the app. Also when we press back button & come out from the app.

- This will close when we close the app by right swipe.

- If we download this episode after some time, notification will be shown in notification bar. This is foreground service.

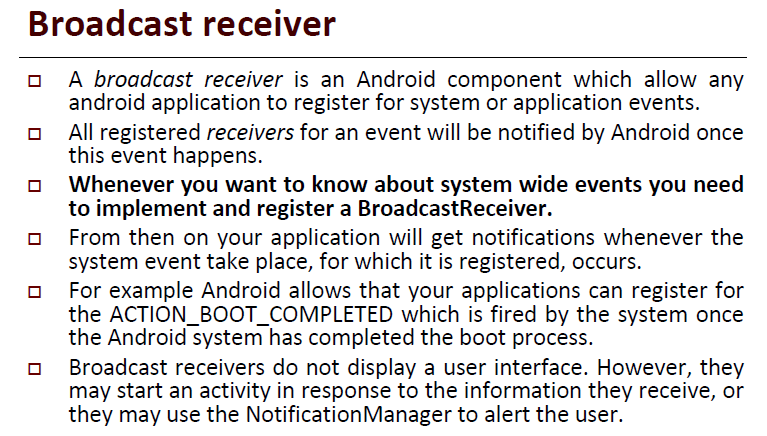
- This will not close when we minimuze the app. Also when we press back button & come out from the app.

- This will even not close when we close the app by right swipe.

- This will only close when we press X button which is used to close all the apps.

**Broadcast Receivers**

Example - <https://www.geeksforgeeks.org/broadcast-receiver-in-android-with-example/>



> We have to write code in onReceive() method for whatever action we want to perform against receiving of notification for our registered broadcast receiver.

> We have to register receiver in Android Manifest (Design time) and programatically in java file (Runtime) both (from android Oreo).

> If we have made Internal Broadcast Receiver(in MainActivity.java), then desgn time (in Manifest) registration is not required. Only runtime(in java file) registration is required.

**INTENTS**

> An Intent object is an abstract description of an operation to be performed.

> Types of Intents :

1) Explicit Intents : It is used when you know the name of activity that you want to launch.

Intent **ExplicitIntent** = **new** Intent(ctx, MyContactActivity.**class**);

2) ImplicitIntents : It is used when you tell the system what do you want to perform and system will find suitable activity for your task.

It includes Action and Data.

Intent **ImplicitIntent** = **new** Intent(Intent.***ACTION\_CALL***, Uri.*parse*(**"tel:"** + **value1**));

----------------------------------------------------------------------------------------------------------------------

**NOTIFICATIONS IN ANDROID**

Example - <https://www.geeksforgeeks.org/notifications-in-kotlin/>

> Steps to Create Notification :

> If android version is Oreo (Android 8, API level 26) and higher, it is necessary to create notification channel. So first check for version as per below;

if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.O) { }

1. Create Notification Channel :

NotificationChannel notificationChannel = new NotificationChannel(channel\_id , channel\_name, NotificationManager.IMPORTANCE\_HIGH);

notificationChannel.setDescription(“Description”);

NotificationManager notificationManager = (NotificationManager) getSystemService(Context.NOTIFICATION\_SERVICE);

notificationManager.createNotificationChannel(notificationChannel);

1. Create Notification :

NotificationCompat.Builder builder = new NotificationCompat.Builder(this, channel\_id)  
        .setSmallIcon(R.drawable.notification\_icon)  
        .setContentTitle(textTitle)  
        .setContentText(textContent)  
        .setPriority(NotificationCompat.PRIORITY\_DEFAULT);

1. Send Notification :

/ notificationId is a unique int for each notification that you must define  
notificationManager.notify(notificationId, builder.build());

**SENSORS IN ANDROID**

Sensors are the devices that contains properties to detect changes in the environment such as light, proximity, rotation, movements, magnetic fields and much more.

> Different types of sensors like accelerometer, ambient light sensor, GPS sensor, compass, proximity sensor, pressure sensor, gyroscope etc are behind the smartphones.

## Categories of Sensors :

The Android platform supports three broad categories of sensors:

**Motion Sensors**  
These sensors track movement; they include accelerometers, gyroscopes and gravity sensors. They provide data on forces like acceleration and rotation that act on the sensor’s three-dimensional axes.

**Environmental Sensors**  
Barometers and thermometers are types of sensors that access environmental metrics. These sensors monitor environmental variables like air pressure and temperature.

**Position Sensors**  
Magnetometer and orientation sensors help determine the physical position of a device.

Each of these categories represents many specific sensors that are available on a device. You will go through them next.

> Types of some commonly used sensors :

**Proximity Sensor**

Detects when an object is near to the phone. Most commonly used to sense when a phone is held up to the users ear to turn off the display. This saves both battery life and prevents accidental screen touches.

They comes under Position Sensors.

**Accelerometer and gyroscope**

Accelerometers in mobile phones are used to detect the orientation of the phone. The gyroscope, or gyro for short, adds an additional dimension to the information supplied by the accelerometer by tracking rotation or twist.

An accelerometer measures linear acceleration of movement, while a gyro on the other hand measures the angular rotational velocity. Both sensors measure rate of change; they just measure the rate of change for different things.

They comes under Motion Sensors.

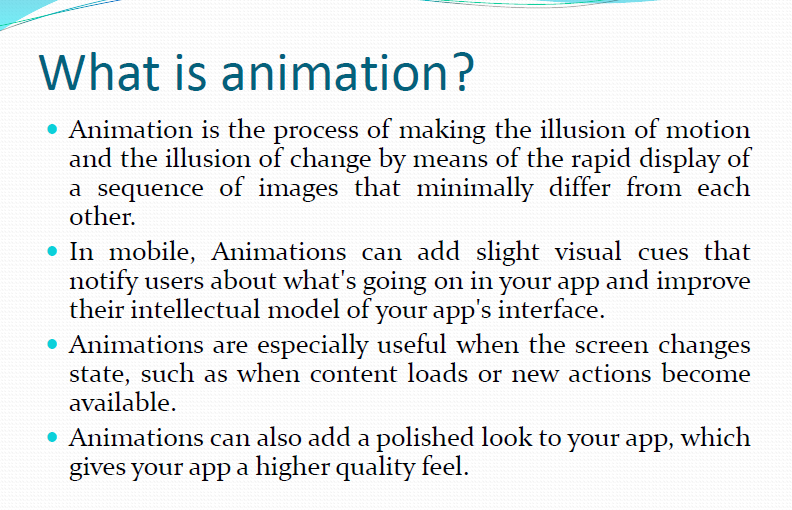
## List of Sensors :

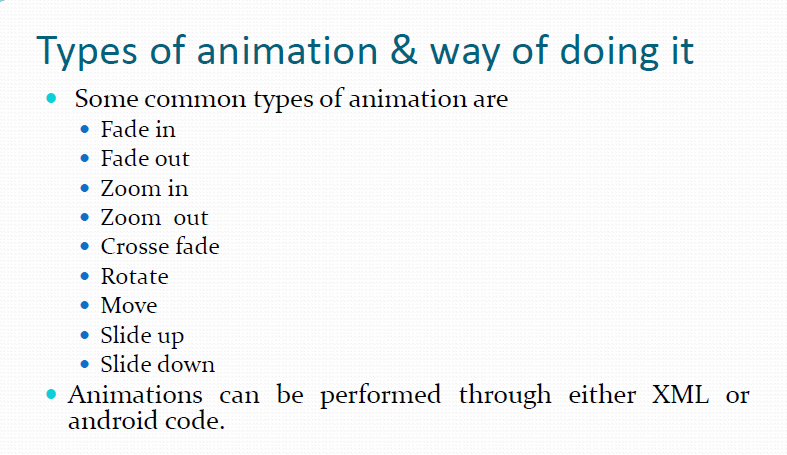
Android SDK provides you with a list of various types of sensors that you can use in your app. The availability of these sensors may vary from device to device.

Here’s a quick rundown of each sensor:

* **TYPE\_ACCELEROMETER**  
  **Type**: Hardware  
  Computes the acceleration in m/s2 applied on all three axes (x, y and z), including the force of gravity.
* **TYPE\_AMBIENT\_TEMPERATURE**  
  **Type**: Hardware  
  Monitors the temperature of the surroundings in degrees Celsius.
* **TYPE\_GRAVITY**  
  **Type**: Software or Hardware  
  Computes the gravitational force in m/s2 applied on all three axes (x, y and z).
* **TYPE\_GYROSCOPE**  
  **Type**: Hardware  
  Computes the rate of rotation in rad/s around each of the three axes (x, y and z).
* **TYPE\_LIGHT**  
  **Type**: Hardware  
  Evaluates the light around a surrounding in lx units.
* **TYPE\_LINEAR\_ACCELERATION**  
  **Type**: Software or Hardware  
  Computes the acceleration force in m/s2 applied on all three axes (x, y and z), excluding the force of gravity.
* **TYPE\_MAGNETIC\_FIELD**  
  **Type**: Hardware  
  Computes the geomagnetic field for all three axes in tesla (μT).
* **TYPE\_ORIENTATION**  
  **Type**: Software  
  Computes the degree of rotation around all three axes.
* **TYPE\_PRESSURE**  
  **Type**: Hardware  
  Computes the air pressure in hPa or mbar.
* **TYPE\_PROXIMITY**  
  **Type**: Hardware  
  Computes the proximity of the device’s screen to an object in centimeters.
* **TYPE\_RELATIVE\_HUMIDITY**  
  **Type**: Hardware  
  Computes the humidity of the surrounding air as a percentage (%).
* **TYPE\_ROTATION\_VECTOR**  
  **Type**: Software or Hardware  
  Computes the orientation of a device by the device’s rotation vector.
* **TYPE\_TEMPERATURE**  
  **Type**: Hardware  
  Monitors the temperature of the surroundings in degrees Celsius. In API 14, the **TYPE\_AMBIENT\_TEMPERATURE** sensor replaced this sensor.

**ANIMATIONS IN ANDROID**





**OOPs Concepts**

>> OOPs (Object-Oriented Programming System) :

**> Object** means a real-world entity such as a pen, chair, table, computer, watch, etc. **Object-Oriented Programming** is a methodology or paradigm to design a program using classes and objects. It simplifies software development and maintenance by providing some concepts: (Short to remember- Object,class, EPIA)

* [Object](https://www.javatpoint.com/object-and-class-in-java)
* Class
* [Inheritance](https://www.javatpoint.com/inheritance-in-java)
* [Polymorphism](https://www.javatpoint.com/runtime-polymorphism-in-java)
* [Abstraction](https://www.javatpoint.com/abstract-class-in-java)
* [Encapsulation](https://www.javatpoint.com/encapsulation)

1) Object :

> An Object can be defined as an instance (example, illustration) of a class. An object contains an address and takes up some space in memory.

> Any entity that has state (existance of something) and behavior is known as an object. For example, a chair, pen, table, keyboard, bike, etc. It can be physical or logical.

2) Class :

> Class is a group of variables of different data types and a group of methods.

> Collection of objects is called class. It is a logical entity.

> A class can also be defined as a blueprint (guide for making something) from which you can create an individual object. Class doesn't consume any space.

3) Inheritance :

> *When one object or class acquires all the properties and behaviors of a parent object or class*, it is known as inheritance. It provides code reusability. It is used to achieve runtime polymorphism.

## **Q) Why multiple inheritance is not supported in java?**

To reduce the complexity and simplify the language, multiple inheritance is not supported in java.

Consider a scenario where A, B, and C are three classes. The C class inherits A and B classes. If A and B classes have the same method and you call it from child class object, there will be ambiguity to call the method of A or B class.

Since compile-time errors are better than runtime errors, Java renders compile-time error if you inherit 2 classes. So whether you have same method or different, there will be compile time error.

4) Polymorphism :

> If *one task is performed in different ways*, it is known as polymorphism.

> In Java, we use method overloading and method overriding to achieve polymorphism.

1. [Method Overloading in Java](https://beginnersbook.com/2013/05/method-overloading/) – This is an example of compile time (or static polymorphism) because which method is to be called is determined by the arguments we pass while calling methods. This happens at  compile time so this type of polymorphism is known as compile time polymorphism.  
2. [Method Overriding in Java](https://beginnersbook.com/2014/01/method-overriding-in-java-with-example/) – This is an example of runtime time (or dynamic polymorphism).

**Runtime polymorphism** or **Dynamic Method Dispatch** is a process in which a call to an overridden method is resolved at runtime rather than compile-time.

In this process, an overridden method is called through the reference variable of a superclass. The determination of the method to be called is based on the object being referred to by the reference variable.

5) Abstraction :

*> Hiding internal details and showing functionality* is known as abstraction. For example phone call, we don't know the internal processing.

> In Java, we use abstract class and interface to achieve abstraction.

6) Encapsulation :

> Binding (or wrapping) code and data together into a single unit are known as encapsulation*. For example, a capsule, it is wrapped with different medicines.*

*> A java class is the example of encapsulation. Java bean is the fully encapsulated class because all the data members are private here.*

> Advantages of OOPs over POP :

1) OOPs makes development and maintenance easier, whereas, in a procedure-oriented programming language, it is not easy to manage if code grows as project size increases.

2) OOPs provides data hiding, whereas, in a procedure-oriented programming language, global data can be accessed from anywhere.

**Activity Lifecycle**

> Android Activity Lifecycle is controlled by 7 methods of android.app.Activity class. The android Activity is the subclass of ContextThemeWrapper class.

An activity is the single screen in android. It is like window or frame of Java.

By the help of activity, you can place all your UI components or widgets in a single screen.

> Let's see the 7 lifecycle methods of android activity :

|  |  |
| --- | --- |
| **Method** | **Description** |
| **onCreate** | called when activity is first created. |
| **onStart** | called when activity is becoming visible to the user. |
| **onResume** | called when activity will start interacting with the user in foreground after onStart. |
| **onPause** | Activity is partially obscured by another activity or when u minimize the app. |
| **onStop** | The activity is completely hidden and not visible to the user. |
| **onRestart** | called after your activity is stopped, prior to start. |
| **onDestroy** | called before the activity is destroyed. |



> Now let’s see the situations where the life cycle methods and states will occur :

* **When you open the app it will go through below states:**

onCreate() –> onStart() –>  onResume()

* **When you press the back button and exit the app**

onPaused() — > onStop() –> onDestory()

* **When you press the home button**

onPaused() –> onStop()

* **After pressing the home button, again when you open the app from a recent task list**

onRestart() –> onStart() –> onResume()

* **After dismissing the dialog (eg. Alert dialog msg) or back button from the dialog**

onResume()

* **If a phone is ringing and user is using the app**

onPause() –> onResume()

* **After the call ends**

onResume()

* **When your phone screen is off**

onPaused() –> onStop()

* **When your phone screen is turned back on**

onRestart() –> onStart() –> onResume()

**Android**

> Latest Android Versions & its API levels :

|  |  |  |
| --- | --- | --- |
| **Name** | **Version number(s) API Level** | **Initial stable release date** |
| **Pie** | 9 28 | August 6, 2018 |
| **Android 10(Q)** | 10 29 | September 3, 2019 |
| Android 11(R) | 11 30 | September 8, 2020 |
| Android 12 | 12 31 | TBA |

> What is Android ?

**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(native libraries)** 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.

> Android Architecture :

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

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

## **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.

**Storage**

> Shared Preferences :

> If you have a relatively small private primitive data collection of key-values that you'd like to save, you should use the [SharedPreferences](https://developer.android.com/reference/android/content/SharedPreferences) APIs.

> **Android** stores **Shared Preferences** settings as XML **file** in shared\_prefs folder under DATA/data/{application package} directory. The DATA folder can be obtained by calling Environment. getDataDirectory() .

> Internal Storage :

> We are able to save or read data from the device internal memory. FileInputStream and FileOutputStream classes are used to read and write data into the file.

Here, we are going to read and write data to the internal storage of the device.

> External Storage :

> Like internal storage, we are able to save or read data from the device external memory such as sdcard. The FileInputStream and FileOutputStream classes are used to read and write data into the file.

> [SQLite Databases](file:///E:\Android%20Studio\Android\sdk\docs\guide\topics\data\data-storage.html#db) :

> Store structured data in a private database.

> [Network Connection](file:///E:\Android%20Studio\Android\sdk\docs\guide\topics\data\data-storage.html#netw) :

> Store data on the web with your own network server.

## **Using Shared Preferences**

The [SharedPreferences](file:///E:\Android%20Studio\Android\sdk\docs\reference\android\content\SharedPreferences.html) class provides a general framework that allows you to save and retrieve persistent key-value pairs of primitive data types. You can use [SharedPreferences](file:///E:\Android%20Studio\Android\sdk\docs\reference\android\content\SharedPreferences.html) to save any primitive data: booleans, floats, ints, longs, and strings. This data will persist(continue) across user sessions (even if your application is killed).

### User Preferences

Shared preferences are not strictly for saving "user preferences," such as what ringtone a user has chosen. If you're interested in creating user preferences for your application, see [PreferenceActivity](file:///E:\Android%20Studio\Android\sdk\docs\reference\android\preference\PreferenceActivity.html), which provides an Activity framework for you to create user preferences, which will be automatically persisted (using shared preferences).

To get a [SharedPreferences](file:///E:\Android%20Studio\Android\sdk\docs\reference\android\content\SharedPreferences.html) object for your application, use one of two methods:

* [getSharedPreferences()](file:///E:\Android%20Studio\Android\sdk\docs\reference\android\content\Context.html#getSharedPreferences(java.lang.String, int)) - Use this if you need multiple preferences files identified by name, which you specify with the first parameter.
* [getPreferences()](file:///E:\Android%20Studio\Android\sdk\docs\reference\android\app\Activity.html#getPreferences(int)) - Use this if you need only one preferences file for your Activity. Because this will be the only preferences file for your Activity, you don't supply a name.

To write values:

1. Call [edit()](file:///E:\Android%20Studio\Android\sdk\docs\reference\android\content\SharedPreferences.html#edit()) to get a [SharedPreferences.Editor](file:///E:\Android%20Studio\Android\sdk\docs\reference\android\content\SharedPreferences.Editor.html).
2. Add values with methods such as [putBoolean()](file:///E:\Android%20Studio\Android\sdk\docs\reference\android\content\SharedPreferences.Editor.html#putBoolean(java.lang.String, boolean)) and [putString()](file:///E:\Android%20Studio\Android\sdk\docs\reference\android\content\SharedPreferences.Editor.html#putString(java.lang.String, java.lang.String)).
3. Commit the new values with [commit()](file:///E:\Android%20Studio\Android\sdk\docs\reference\android\content\SharedPreferences.Editor.html#commit())

To read values, use [SharedPreferences](file:///E:\Android%20Studio\Android\sdk\docs\reference\android\content\SharedPreferences.html) methods such as [getBoolean()](file:///E:\Android%20Studio\Android\sdk\docs\reference\android\content\SharedPreferences.html#getBoolean(java.lang.String, boolean)) and [getString()](file:///E:\Android%20Studio\Android\sdk\docs\reference\android\content\SharedPreferences.html#getString(java.lang.String, java.lang.String)).

Example to Add data using SharedPreferences :

SharedPreferences sp = getSharedPreferences(**"datafile"**, ***MODE\_PRIVATE***); *// This will create an XML file "datafile", in which data will be stored.*SharedPreferences.Editor ed = sp.edit(); *// Editor object created. It will be needed to write and delete data. It will not needed to read data.*ed.putString(**"email"**, **txttext**.getText().toString()); *// Data of edittext is inserted into above "datafile".*ed.commit(); *// It is must. Without this, operation will not be completed.*

Example to View data using SharedPreferences :

SharedPreferences sp = getSharedPreferences(**"datafile"**, ***MODE\_PRIVATE***);  
  
**if**(sp.contains(**"email"**))  
 **lblshow**.setText(sp.getString(**"email"**,**""**));  
**else** Toast.*makeText*(**ctx**, **"Data not found"**, Toast.***LENGTH\_LONG***).show();

Example to Delete data using SharedPreferences :

SharedPreferences sp = getSharedPreferences(**"datafile"**, ***MODE\_PRIVATE***);  
SharedPreferences.Editor ed = sp.edit();  
  
**if**(sp.contains(**"email"**))  
{  
 ed.remove(**"email"**);  
 ed.commit();  
 **lblshow**.setText(**""**);  
 Toast.*makeText*(**ctx**, **"Data deleted successfully"**, Toast.***LENGTH\_LONG***).show();  
}

> Link to save ArrayList in Shared Preferences and Retrive it in RecyclerView :

<https://www.geeksforgeeks.org/how-to-save-arraylist-to-sharedpreferences-in-android/>

or

<https://codinginflow.com/tutorials/android/save-arraylist-to-sharedpreferences-with-gson>

**General Topics**

## Software Development Lifecycle :

Different phases of the software development cycle

Software Development Life Cycle

* [Requirement Phase](https://www.javatpoint.com/software-development-life-cycle#requirement-phase)
* [Design Phase](https://www.javatpoint.com/software-development-life-cycle#design-phase)
* [Build /Development Phase](https://www.javatpoint.com/software-development-life-cycle#build-development-phase)
* [Testing Phase](https://www.javatpoint.com/software-development-life-cycle#testing-phase)
* [Deployment/ Deliver Phase](https://www.javatpoint.com/software-development-life-cycle#deployment-deliver-phase)
* [Maintenance](https://www.javatpoint.com/software-development-life-cycle#maintenance)

**> Difference between ListView and RecycleView :**

| **RecyclerView** | **ListView** |
| --- | --- |
| In a RecyclerView, your items will have a specific View, these Views are placed inside ViewHolders. Each ViewHolder can have up to one View and and the ViewHolder is used to display a particular View of an item within your RecyclerView.  For performance reasons, the RecyclerView will reuse ViewHolders for displaying Views by recycling ViewHolders that contain Views that shift offscreen with Views that shift onscreen when scrolling occurs. | We requires separate view or viewgroup cell for each items in list. |
| Mandatory ViewHolder pattern providing better performance | ViewHolder pattern can be supported with customisation |
| The RecyclerView uses a LayoutManager to determine how the Views will be placed within the RecyclerView. The different LayoutManagers supported by RecyclerView include:   * LinearLayoutManager, a vertical or horizontal list of items * GridLayoutManager, a grid of items that are all the same size.( Like in gallery). * StaggeredGridLayoutManager, a grid of items that supports different sizes | It supports Vertical list only. |
| * Supports notifyDataSetChange(), if the entire data set has changed. Also supports * notifyItemInserted(), notifyItemRemoved(), notifyItemChanged() when item gets added, item gets removed or item gets changed | * Only supports notifyDataSetChange(), if the entire data set has changed. |
| ItemAnimator makes it easy to add animations | Very complex to implement |
| It is used when you have a large data set you want to show in a list. | It is used when you have a small list of items to display. |
| Less memory consuption. | More memory consuption as the items in the list increases. |
| Dividers between items not shown by default. Use ItemDecorations to add margin and draw on or under an Item View. | Dividers between items shown by default. Item decoration requires customisation for ListView. |

Above content is taken from below link :

<https://learntodroid.com/what-are-the-differences-between-recyclerview-and-listview/>

**> Changes in Android 11 :**

1) Greater control over conversation notifications :

Android 11 has "promoted" conversations. Now all conversations from text messages and other chat apps appear in their own section, separate from other notifications. Moreover, [you have control over the priority of these conversations](https://www.businessinsider.com/how-to-set-conversation-priority-on-android) - you can allow them to appear normally, set certain ones to always appear at the top of the list, or make them "silent," which turns off notification sounds and makes them appear at the bottom of the list.

2) Recall cleared notifications with notification history :

If you clear a notification and instantly regret it because you don't recall what it said, you'll appreciate the new notification history in Android 11. Found in Settings, it [lets you see all notifications you cleared in the last 24 hours](https://www.businessinsider.com/how-to-see-android-notification-history).

3)Chat Bubbles :

Some conversations can also be turned into [Bubbles](https://www.androidpolice.com/2020/02/19/bubble-notifications-are-no-longer-hidden-in-android-11/), which are floating circular buttons that maximize or minimize the chat in a floating window when tapped. During this we can continue to work with our current app.

1. Built-in Screen Recording with sound.
2. Voice Access gets smarter :

Android's Accessibility features include Voice Access, a feature that lets you control the phone's interface via voice command. In the past, there were many situations in which you had to pick options off the screen using numbers, but in Android 11, the OS has a better understanding of the context of your commands. As a result, most of the need of choosing options via numbers is gone.

1. Dynamic media controls :

 if you have more than one media app running, you can swipe side to side to see the controls for each one.

Switch from your headphones to your speaker without missing a beat. Tap to hear your tunes or watch video on your TV. With Android 11, you can quickly change the device that your media plays on.

1. Android Auto(Cars) now works wirelessly with devices running Android 11—so you can bring the best of your phone in your car on every drive.

8) Better Control on Permissions :

> One Time Permission

> Permission auto reset

If you haven't used an app in a while, you may not want it to keep accessing your data. So Android will reset permissions for your unused apps. You can always turn permissions back on.

9)A more useful power button menu :

Press and hold the power button to see an enhanced screen with access to your wallet, emergency calling, and smart home devices. Set the temperature to chill, then dim your lights. All from a single spot on your phone.3

10) Pin apps in share menu.

11) Security updates from google play.

12) Scheduled dark mode.

> Links for above content (Changes in Android 11): There are pictures and videos in below links. So refer it, if you want to get better understanding.

<https://www.android.com/intl/en_in/android-11/>

<https://www.businessinsider.in/tech/how-to/the-11-best-new-features-of-android-11-that-make-your-phone-more-dynamic-and-easier-to-use/articleshow/81712016.cms>

<https://www.androidpolice.com/2020/09/16/11-new-android-11-features-you-should-know-about/>

> What is a developer preview?

The **Developer Preview** builds provide an early test and development environment you can use to try new APIs, identify compatibility issues in your app, and plan migration or feature work needed to target the new platform.

> What is alpha and beta version?

**Alpha** software is not thoroughly tested by the developer(supplier) before it is released to customers. ... At this time, the software is said to be feature complete. A **beta** test is carried out following acceptance **testing** at the supplier's site (**alpha** test) and immediately prior to general release of the software as a product.

> What is difference between beta version and normal version?

An app is the app that is ready **for** production to all its users, the phase comes after the **beta** test. The app is available to all the users available in market. **Beta** apps are not available to all its user it is available to the group **of** people or who like to enroll **for beta** testing.

**> Java Arrays :**

> **Java array** is an object which contains elements of a similar data type.

> Array in Java is index-based, the first element of the array is stored at the 0th index, 2nd element is stored on 1st index and so on.

### **Advantages**

* **Code Optimization:** It makes the code optimized, we can retrieve or sort the data efficiently.
* **Random access:** We can get any data located at an index position.

### **Disadvantages**

* **Size Limit:** We can store only the fixed size of elements in the array. It doesn't grow its size at runtime. To solve this problem, collection framework is used in Java which grows automatically.

**Example : int** a[]={33,3,4,5};//declaration, instantiation and initialization

System.out.println(a[2]); // This will print 4

### > **Why is the Java main method static in java ?**

> When java runtime starts, there is no object of the class present. That’s why the main method has to be static so that JVM can load the class into memory and call the main method. If the main method won’t be static, JVM would not be able to call it because there is no object of the class is present. Let’s see what happens when we remove static from java main method.

> If it were a non-static method, [JVM](https://www.javatpoint.com/jvm-java-virtual-machine) creates an object first then call main() method that will lead the problem of extra memory allocation.

**> Java Collections :**

> The **Collection in Java** is a framework that provides an architecture to store and manipulate the group of objects.

> Java ArrayList :

- Java **ArrayList** class uses a dynamic [*array*](https://www.javatpoint.com/array-in-java) for storing the elements. It is like an array, but there is no size limit. We can add or remove elements anytime. So, it is much more flexible than the traditional array.

Example :

ArrayList<String> list=**new** ArrayList<String>();//Creating arraylist

      list.add("Mango");//Adding object in arraylist

      list.add("Apple");

      list.add("Banana");

      list.add("Grapes");

      //Printing the arraylist object

      System.out.println(list);

// This will print following without Iterator.

[Mango, Apple, Banana, Grapes]

//Traversing list through Iterator

  Iterator itr=list.iterator();//getting the Iterator

**while**(itr.hasNext()){//check if iterator has the elements

   System.out.println(itr.next());//printing the element and move to next

// This will print following.

Mango

Apple

Banana

Grapes

**> Content Providers :**

In android, **Content Provider** will act as a central repository to store the data of the application in one place and make that data available for different applications to access whenever it’s required.

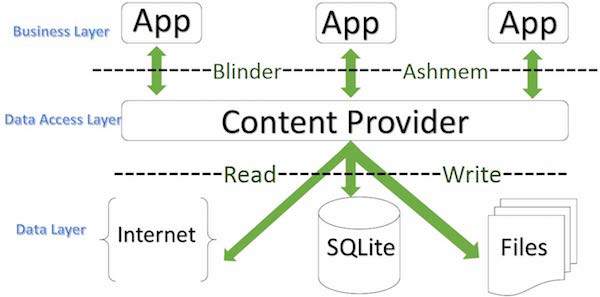
In android, we can configure Content Providers to allow other applications securely access and modify our app data based on our requirements.

Generally, the **Content Provider** is a part of an android application and it will act like more like a relational database to store the app data. We can perform multiple operations like insert, update, delete and edit on the data stored in content provider using **insert()**, **update()**, **delete()** and **query()** methods.

In android, we can use content provider whenever we want to share our app data with other apps and it allow us to make a modifications to our application data without effecting other applications which depends on our app.

In android, the content provider is having different ways to store app data. The app data can be stored in a SQLite database or in files or even over a network based on our requirements. By using content providers we can manage data such as audio, video, images and personal contact information.

We have different type of access permissions available in content provider to share the data. We can set a restrict access permissions in content provider to restrict data access limited to only our application and we can configure different permissions to read or write a data.



**> What is ReactNative? :**

React Native is an open-source mobile application framework created by Facebook, Inc in 2015. It is used to develop applications for Android, Android TV, iOS, macOS, tvOS, Web, Windows and UWP by enabling developers to use React's framework along with native platform capabilities.

There is no **need** to know the languages used for platform-specific **application development**, as **React Native** uses only JavaScript, probably the most popular programming language right now, for the **development** of mobile applications.

> What is React ?

React is an open-source, front end, JavaScript library for building user interfaces or UI components.

**> What is GitHub? How to use it ? :**

[**https://guides.github.com/activities/hello-world/**](https://guides.github.com/activities/hello-world/)

> What is Version Control in Github ?

**Version control** allows you to keep track of your work and helps you to easily explore the changes you have made, be it data, coding scripts, notes, etc. ... With **version control** software such as Git, **version control** is much smoother and easier to implement.

**What is Gradle?**

Gradle is a build automation tool known for its flexibility to build software. A build automation tool is used to automate the creation of applications. The building process includes compiling, linking, and packaging the code. The process becomes more consistent with the help of build automation tools.

**Why is Gradle needed?**

Every Android project needs a gradle for generating an apk from the *.java* and *.xml* files in the project. Simply put, a gradle takes all the source files (java and XML) and apply appropriate tools, e.g., converts the java files into dex files and compresses all of them into a single file known as apk that is actually used.

**> What is Gradle Files? :**

**gradle files** are the main script **files** for automating the tasks in an **android** project and are used by the **Gradle** for generating the APK from the source **files**.

**> What is Build Process? :**

The build process involves many tools and processes that convert your project into an Android Application Package (APK).

**What are push notifications?**

A push notification is a message that pops up on a mobile device. App publishers can send them at any time; users don’t have to be in the app or using their devices to receive them. They can do a lot of things; for example, they can show the latest sports scores, get a user to take an action, such as downloading a coupon, or let a user know about an event, such as a flash sale.

Push notifications look like SMS text messages and mobile alerts, but they only reach users who have installed your app. Each mobile platform has support for push notifications — iOS, Android, Fire OS, Windows and BlackBerry all have their own services.

**What are in-app notifications?**

In-app notifications are messages that app creators can send to users within their app. They’re commonly used to direct users toward points of interest to boost usage, retention, and lifetime value (LTV). Used properly, in-app notifications help users find what they’re looking for and increase their satisfaction. Used improperly, they might encourage users to stop using your app.

The content of these in-app notifications could include asking users to rate the app, asking users for push notification opt-ins, asking users for feedback etc..

## Compare in-app notifications and push notifications:

|  |  |  |
| --- | --- | --- |
|  | **In-app notification** | **Push notification** |
|  | Inside of app | Outside of app |
|  | Guides users | Brings users to the app |
|  | Only works when app is open | Works any time |
|  | Captive audience | Potentially disengaged audience |
|  | Product teams control whether users can disable them | Users always reserve the right to disable them |

**Steps - How to manually make widget(on home screen) ?**

- Right click on package > New > Widget > App Widget.

- Create the layout resourse file for widget.

- Create new java class which extends AppWidgetProvider.

- Override onUpdate() method.

- Register a broadcast receiver in Manifest for app widget update.

**What is MVP ?**

Model–view–presenter (**MVP**) is a derivation of the model–view–controller (MVC) architectural pattern which mostly used for building user interfaces.

MVVM stands for **Model**, **View**, **ViewModel**.

**Why do we need these patterns?**  
Adding everything in a Single Activity or Fragment would lead to problems in testing and refactoring the code. Hence, the use of separation of code and clean architecture is recommended.

| **Codename** | **Version** | **API level/NDK release** |
| --- | --- | --- |
| Android11 | 11 | API level 30 |
| Android10 | 10 | API level 29 |
| Pie | 9 | API level 28 |
| Oreo | 8.1.0 | API level 27 |
| Oreo | 8.0.0 | API level 26 |
| Nougat | 7.1 | API level 25 |
| Nougat | 7.0 | API level 24 |
| Marshmallow | 6.0 | API level 23 |
| Lollipop | 5.1 | API level 22 |
| Lollipop | 5.0 | API level 21 |
| KitKat | 4.4 - 4.4.4 | API level 19 |
| Jelly Bean | 4.3.x | API level 18 |
| Jelly Bean | 4.2.x | API level 17 |
| Jelly Bean | 4.1.x | API level 16 |
| Ice Cream Sandwich | 4.0.3 - 4.0.4 | API level 15, NDK 8 |
| Ice Cream Sandwich | 4.0.1 - 4.0.2 | API level 14, NDK 7 |
| Honeycomb | 3.2.x | API level 13 |
| Honeycomb | 3.1 | API level 12, NDK 6 |
| Honeycomb | 3.0 | API level 11 |
| Gingerbread | 2.3.3 - 2.3.7 | API level 10 |
| Gingerbread | 2.3 - 2.3.2 | API level 9, NDK 5 |
| Froyo | 2.2.x | API level 8, NDK 4 |
| Eclair | 2.1 | API level 7, NDK 3 |
| Eclair | 2.0.1 | API level 6 |
| Eclair | 2.0 | API level 5 |
| Donut | 1.6 | API level 4, NDK 2 |
| Cupcake | 1.5 | API level 3, NDK 1 |
| (no codename) | 1.1 | API level 2 |
| (no codename) | 1.0 | API level 1 |

**What is API Level?**

**API Level** is an integer value that uniquely identifies the framework **API** revision offered by a **version** of the **Android** platform. The **Android** platform provides a framework **API** that applications can use to interact with the underlying **Android** system.

### **What is Runtime permissions ?**

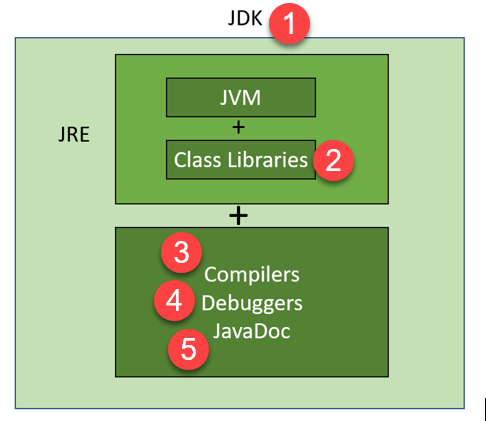
Runtime permissions, also known as dangerous permissions. Dangerous permissions are permissions which could potentially affect the user's privacy or the device's operation. The user must explicitly agree to grant those permissions. These include accessing the camera, contacts, call, location, microphone, sensors, SMS, and storage.

If you declare any [dangerous permissions](https://developer.android.com/guide/topics/permissions/overview#dangerous_permissions), and if your app is installed on a device that runs Android 6.0 (API level 23, Marshmallow) or higher, you must request the dangerous permissions at runtime (in addition of design time).

**What is Admin Panel ?**

The administration panel (or just Admin) is the interface used by all web based companies to manipulate their App's data and follow their business metrics (KPIs). Operations teams constantly use it to enter and update data in their App.

**Difference betn JVM, JRE & JDK ?**



**JVM :**

JVM (Java Virtual Machine) is an abstract machine. It is called a virtual machine because it doesn't physically exist. It is a specification that provides a runtime environment in which Java bytecode can be converted into machine code. JVM is a part of Java Run Environment (JRE).

The JVM performs the following main tasks:

* Loads code
* Verifies code
* Executes code
* Provides runtime environment

**JRE :**

JRE (Java Runtime Environment) is a software package that provides Java class libraries, Java Virtual Machine (JVM), and other components that are required to run Java applications.

**JDK :**

JDK is an acronym for Java Development Kit. The Java Development Kit (JDK) is a software development environment which is used to develop Java applications and [applets](https://www.javatpoint.com/java-applet). It physically exists.

In addition to JRE, JDK also contains a number of development tools (compilers, JavaDoc, Java Debugger, etc).

**Compilers** – A compiler is a program that translates(compiles) source code written in high-level language(eg. C++) into machine-level language that can be understood by CPU. It has error-checking ability also.

**What is REST and RESTfull API ?**

The short answer is that REST stands for Representational State Transfer. It’s an [architectural pattern](https://blog.ndepend.com/software-architecture-5-patterns-you-need-know/) for creating API(web services). It is a set of rules that developers follow when they create their API.

A RESTful API(web service) is one that implements that pattern.

**What is JSON ?**

JSON (an acronym for **JavaScript** Object Notation) is a lightweight data-interchange format and is most commonly used for client-server **communication**. It's both easy to read/write and language-independent. A JSON value can be another JSON object, array, number, string, boolean (true/false) or null.

**What is JSON Object ?**

JSON objects are written in key/value pairs, surrounded by curly braces {}.

Keys must be strings, and values must be a valid JSON data type (string, number, object, array, boolean or null).

Keys and values are separated by a colon.

Each key/value pair is separated by a comma.

Ex. { "name":"John", "age":30, "car":null }

**> SDK, API, IDE, Framework, Library, Platform- Definitions :**

<https://medium.com/@shashvatshukla/framework-vs-library-vs-platform-vs-api-vs-sdk-vs-toolkits-vs-ide-50a9473999db>

**Platform:** Any hardware or software environment in which a program runs, is known as a platform. Since Java has a runtime environment (JRE) and API, it is called a platform.

A platform is simply the hardware or software for which the piece of software is built. For example, software may be built for a *Windows, OS X, Android, iOS, XBOX One, PS4, etc.*These are all platforms.Since different platforms have different requirements and interface differently to the software, the code you write may not run on all platforms and it is important to know which platforms you are building for.

**Library:**A library refers to code that provides functions that you can call from your own code to deal with common tasks. For example a math library will provide you with common mathematical functionality such as trigonometric or logarithmic functions. Programming languages usually have libraries for all sorts of tasks such as data processing, plotting of graphs, text parsing, etc. Once included, libraries save you the trouble of writing all those functions yourself.

**API:**

> An API is a set of programming instructions and standards for accessing a web tool or database.

> In computing, an application programming interface is an interface that defines interactions between multiple software applications .

> An API is a software intermediary that allows two applications to talk to each other. In other words, an API is the messenger that delivers your request to the provider that you're requesting it from and then delivers the response back to you.

> Some examples of APIs are as below :

* Inserting or retriving data from database.
* Weather Snippets. Google utilizes APIs to display relevant data from user search queries. ...
* Log-in Using XYZ. Taken from Buffer's social login. ...
* Pay with PayPal. Most online stores offer Paypal and other payment methods, utilizing API connections to these services. ...
* Twitter Bots. ...
* Travel Booking.

**IDE:**Short for *Integrated Development Environment*. The IDE is an application which helps you during the process of writing the code itself by automating many useful processes such as debugging, refactoring, code generation, etc. An IDE is just a tool to help programmers, and you may simply use Notepad if you wish. Examples of IDEs include: *Eclipse, IntelliJ IDEA, Netbeans, Visual Studio, etc.*

**SDK:**Short for *Software Development Kit.*This is a complete kit of software development tools for a specific platform. This “kit” can include all sorts of things such as: Libraries, APIs, IDEs, Documentation, etc. For example the Android SDK, which provides everything you may need for Android development.

**Framework:**

A framework, or software framework, is a platform for developing software [applications](https://techterms.com/definition/application). It provides a foundation on which software developers can build programs for a specific [platform](https://techterms.com/definition/platform). For example, a framework may include predefined [classes](https://techterms.com/definition/class) (like diff.widget classes, recyclerview.adapter class, AppCompatActivity class etc) and [functions](https://techterms.com/definition/function) that can be used to process [input](https://techterms.com/definition/input), manage hardware devices, and interact with [system software](https://techterms.com/definition/systemsoftware). This streamlines the development process since programmers don't need to reinvent the wheel each time they develop a new application.

**> Explanation of Online Shopping App project :**

> We have made a simple online shopping app with admin panel using networking. It is just like amazone, flipcart etc. It includes mobile app of user and web based admin panel for admin also.

> In this project we have also used basics of – PHP for development of web services, HTML & Bootstrap for designing web pages of admin panel, MySQL database for storage & manipulation of data, and phpMyAdmin to manage MySQL database funtions by GUI.

> We have used Third Party Libraries like Material library for widget design, Volley libraty for networking, Butterknife for binding the views.

> We have used Firebase for cloud messaging. By this admin can send cloud messages to all the customers.

**User :**

> In which user can Register (email, mobile, password will be stored in customer table), login, select category, select product, see product detail, add the product to cart & wishlist, delete the product from cart & wishlist.

> For User Interface, we have designed the different screens like Register, Login, Dashboard, CategoryContainer, ProductContainer, ProductDetail, Change Password, Forgot Password, Cart, Wishlist, Checkout, Logout etc.

> User can change password, if he forgot password then password recovery email will be sent to his registered email.

> He can proceed to checkout from cart and can place order and bill will be generated in Bill table of database. After that he can Logout.

**Admin Panel:**

> In admin panel we have designed Dynamic Web Pages for Admin Login (email, password), Categories, Products, Orders, Customers, Change Password, Logout etc.

> We have used embedded PHP code in HTML and web services to show the dynamic data in web pages from MySQL database server. We do not require to call the web service and do the JSON parsing here bcoz both admin panel and web service are in server.

**Steps :**

> We have made Tables in like customer, admin, category, product, cart, wishlist, bill etc. in MySQL database with the help of phpMyAdmin.

> We have made many Web Services, for sending and receiving of data between our app and MySQL database server.

> We have used the RecyclerView to show the details dynamically.

> We have done the programming in container java files to send the request to MySQL server through web service. The response is received in JSON format. JSON parsing is done to obtain the required details dynamically.

> Then this data will be sent to related Adapter file and data will be shown as per related row.xml file design in recycler view.

> What is Cursor ?

> The basic purpose of a Cursor class object is to point to a single row of the result fetched by the query. We load the row pointed by the cursor object. By using cursor we can save lot of ram and memory.

> When we call 'Movetonext' method it keeps going to the next row.

> What is Primary Key ?

A primary key is a field in a table which uniquely identifies each row/record in a database table. Primary keys must contain unique values. A primary key column cannot have NULL values.

A table can have only one primary key, which may consist of single or multiple fields. When multiple fields are used as a primary key, they are called a composite key.

If a table has a primary key defined on any field(s), then you cannot have two records having the same value of that field(s).

> What is Foreign Key ?

A FOREIGN KEY is a field (or collection of fields) in one table, that refers to the PRIMARY KEY in another table. The table with the foreign key is called the child table, and the table with the primary key is called the referenced or parent table.

> How to generate apk file from project ?

> Open Android project > Build > Build Bundles/APKs > Build APKs.

> Now see at bottom right corner in Event log, following message will be shown :

Build APK(s)

APK(s) generated successfully for 1 module:

Module 'app': locate or analye the APK.

> Click on locate in above to see your generated apk file path.

> You can copy or cut & paste this file in your mobile and install the app.

> As per my opinion, we should generate zip file of project rather than apk file. Bcoz size of zip file is very less than apk file. APK file size is same as project folder size.

> To generate apk file for google playstore upload, there is another process to generate apk.

> The advantage of apk file is, if we send it to someone else in mobile, he can directly install & run the apk file in his mobile without opening android studio.

> On other hand, if we send zip file to someone else in mobile, he has to unzip the file, then he has to open android studio and open this unzipped folder in it and then connect his mobile to it and then he can run.

> What happens if we do not take permission of Internet in networking app ?

> The app will not crash (it will open) but no data from internet will be shown.

> And following lines will be found in error mode of Logcat :

NetworkDispatcher.processRequest: Unhandled exception java.lang.SecurityException: Permission denied (missing INTERNET permission?)

java.lang.SecurityException: Permission denied (missing INTERNET permission?)

> Note that Internet permission comes under Normal Permissions (not dangerous permissions). So system will grant this permission automatically, if we have declared it in manifest. There will be no dialogbox shown at app installation time as well as runtime to user.

> What happens if we do not take permission of SEND\_SMS in SMS app ?

> Bcoz sending SMS comes under dangerous permissions, we have to declare it in manifest (design time) and runtime both.

> If we have not declared it in manifest(or not taken runtime permission), app will start but will be stopped when we click on send sms button in app.

> Also following lines will be found in error mode of Logcat :

java.lang.SecurityException: Sending SMS message: uid 10233 does not have android.permission.SEND\_SMS.

> Note that SEND\_SMS permission comes under Dangerous Permissions (not normal permissions). So system will show a dialog box at starting of app, to deny or allow this permission, if we have declared it in manifest.

REVISION FOR INTERVIEW (15.2.2022)

> Android-12 explanation :

> Description of Changes:

<https://www.android.com/intl/en_in/android-12/>

> Short explanation video:

<https://www.youtube.com/watch?v=xKN4kCeln7Q>

> Long explanation video:

<https://www.youtube.com/watch?v=b2013xcK1-o>

> Changes in Android-12 (as per 1st link above) :

> [**Personal**](https://www.android.com/intl/en_in/android-12/#a12-personal)

* + [Material you](https://www.android.com/intl/en_in/android-12/#a12-material-you)
  + [Dynamic colour](https://www.android.com/intl/en_in/android-12/#a12-color-reimagined)
  + [Responsive motion](https://www.android.com/intl/en_in/android-12/#a12-responsive-motion-ui)
  + [Conversation widgets](https://www.android.com/intl/en_in/android-12/#a12-conversation-widgets)
  + [Accessibility improvements](https://www.android.com/intl/en_in/android-12/#a12-accessibility-improvements)

> [**Safe**](https://www.android.com/intl/en_in/android-12/#a12-safe)

* + [Mic & camera indicators and toggles](https://www.android.com/intl/en_in/android-12/#a12-mic-and-camera)
  + [Approximate location permissions](https://www.android.com/intl/en_in/android-12/#a12-location)
  + [Privacy dashboard](https://www.android.com/intl/en_in/android-12/#a12-privacy-ui)
  + [Private Compute Core](https://www.android.com/intl/en_in/android-12/#a12-compute-core)

> [**Effortless**](https://www.android.com/intl/en_in/android-12/#a12-effortless)

* + [Enhanced gaming](https://www.android.com/intl/en_in/android-12/#a12-gaming)
  + [Scrolling screenshots](https://www.android.com/intl/en_in/android-12/#a12-scrolling-screenshots-ui)
  + [Easily switch phones](https://www.android.com/intl/en_in/android-12/#a12-switch-phones-ui)

> Retrofit revision description links :

<https://guides.codepath.com/android/consuming-apis-with-retrofit>

<https://www.journaldev.com/13639/retrofit-android-example-tutorial>

> Volley revision description links :

<https://www.section.io/engineering-education/making-api-requests-using-volley-android/>