# Introduction to Android Development

**Android is one of the most popular operating systems with over 88% of phones running android.**

### XML & Java code

**An XML file is used to build layouts in Android. Layouts are static pages(screens) that can be manipulated using java code (programmatically) Declaring Vis in XML enables us to better separate the presentation of our application from the code that controls its behavior.**

### Installing Android Studio

**Android Studio is an IDE that makes it easy for us to write and build android applications very conveniently we can create emulators (virtual android phones), see previews and build UI layouts using drop features**

**You can go to https://develper. Android.com/studio to download android studio.**

### The Layout Editor

**The layouts editor is used to quickly build layouts by dragging UI elements which is easier to write XML by hand.**

**We can set up different attributes easily using the design mode in the layout editor.**

## Chapter 1: Creating our first APP

**In this chapter, we will create our first android app. I don't expect you to know anything but will walk you through the steps to build your first Android APP.**

### What is an APK?

**An APK is a collection of different files (like code, audio, video, etc.) compiles and bundled into a single file.**

### What is an AVD?

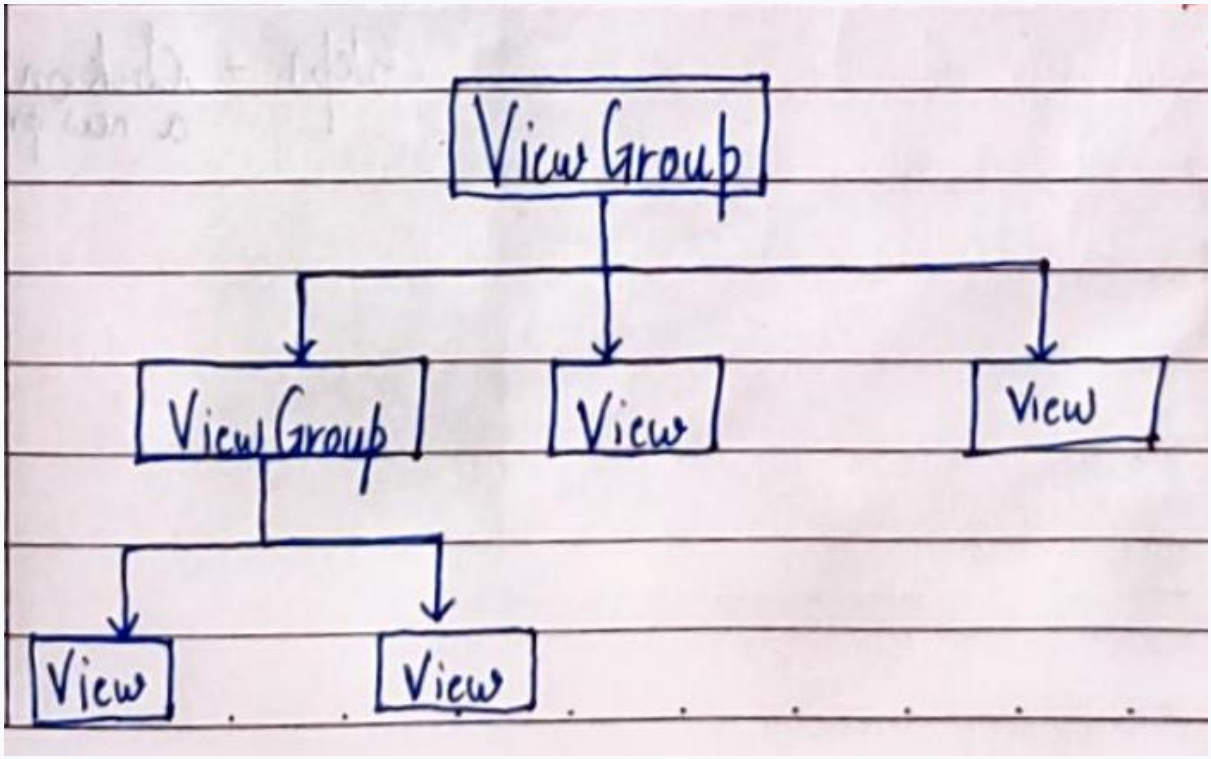
**AVD stands for Android virtual device AVC is an emulator configuration that simulates a physical Android device.**

### Android UI layouts

**View is the base class for widgets** (like **buttons, text fields** etc.)

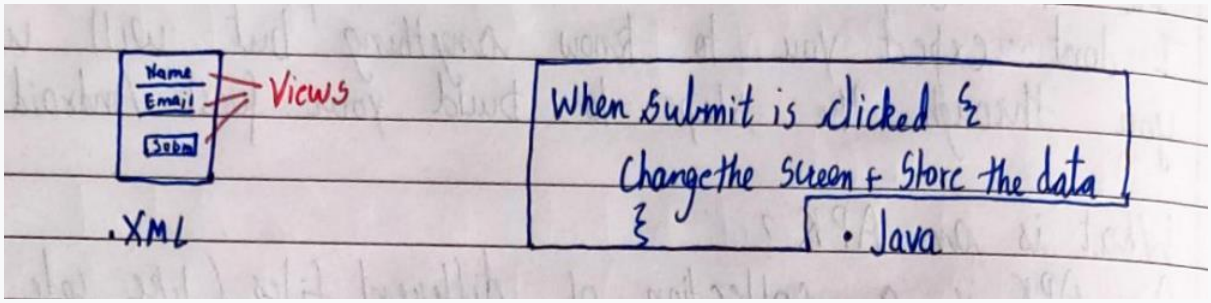
**View = Basic building blocks**

**View group holds view** and **view group**. (just like a box can hold objects and more boxes)




**XML vs Java in Android**

**XML is the skeleton code** that describes the **UId layout java drives this XML**




**Important Notes**

1. **Android studio is a resource hangry program. You need to have patience while using it.**
2. **Sometimes Android studio might download files from the internet, so keep your wifi Hotspot ready**
3. **In very rare cases, your firewall might block android studio.**

4. If your computer is slow, use USB debugging to use your phone as an AVD replacement
5. If you are using an old PC, make sure that the virtualization is turned on

### Creating our unit converter app

We will now create our unit convert application using the Android Studio

### The R. java file

Android R.java file contains resource IDs for all the resources we can use to access views from our java file.

```
Button = find view by Id (R.id.mybotton);
```

### Adding Event listeners

We can add click listeners by using set on click listener method as follows:

```
button.SetonClickListener(new view.OnClickListener())
   @ override
   public void on click (view v) {
   // Action here (this is performed when the button is clicked)
}
});
```

## Chapter 1: Practice Set

1. Create an Android app which is capable of display "good morning" to the user
2. Create an app which is capable of adding two numbers entered by the user
3. Create an App which displays the multiplication table of a given number.

## Chapter 2: Java Refresher

Java is an amazing object-oriented programming language we will use java to create android apps.

### The main method

The java program starts executing from here

```
public static void main (string[] args){
// code
}
```

### Printing to the console

## The following code prints “Hello World” to the console :

```
System.out.print ln (“Hello World”);
```

### Variables in java

## Variables are buckets in memory

```
String actionNow;
```

## The string is used to storing a sequence of characters

```
int marks; // int is used to stone numbers
Int value = 7;
```

### Comments

## Comments are used to write text which doesn’t execute.

```
// this is a comment
/* this is a

    Multi-line comment */
```

### String in java

## String = Sequence of characters

```
String name = “Harry”,
```

## Strings are immutable & cannot be changed

### Printing Strings

## We concatenate string like this:

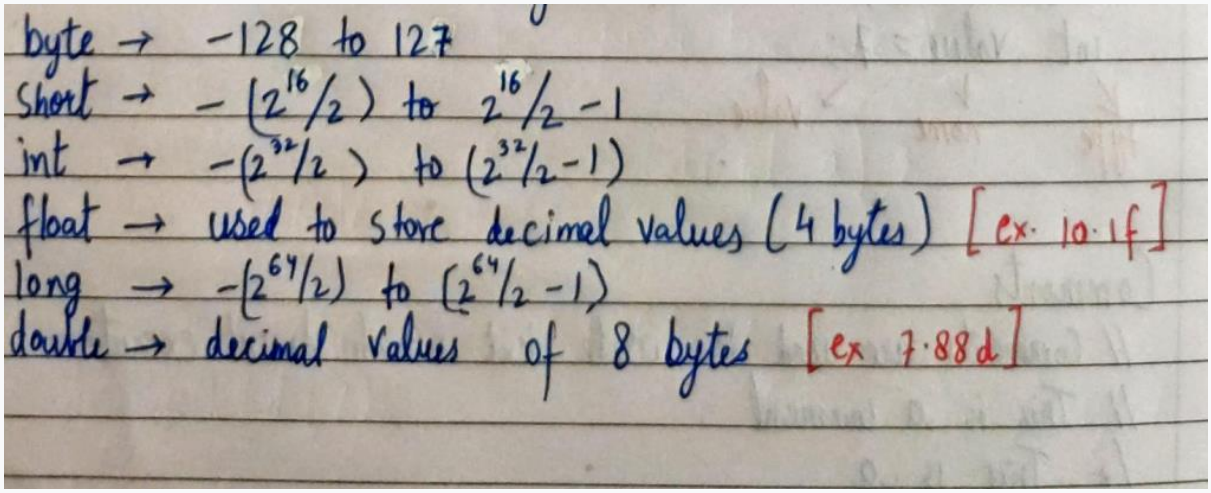
```
System.out.print ln ("Hello," + name );
```

### String methods

```
String name = "Harry"
name.length()
name.to lowerCase ()
name.trim()
```

### Other data types to store numbers

## We can store numbers using




### Booleans

## True or false values

```
boolean isGood = true;
```

### Operators in java

## These are the types of operators in java:

- **Arithmetic operators – a+b, a*b, a/b, a%b**
- **Assignment operators – a= 3, a+ =3**
- **Relational operators – a<b, a>3**
- **Logical operators – (a>3) 88 (b<7)**
- **Unary operators – a++, b—**
- **Bitwise operators - _, <<, 8,1**

### If – else conditionals

**We can use if-else to execute instructions when a condition is true**

```
if (a>3) {
//code
}
else if (a>1) {
            //code
            }
else {
       //code
         }
```

**We can use if inside an if, if inside an else, and Booleans inside if (as conditions)**

**Loops in Java**

1. **For loop: the syntax of a for loop looks like this:**

```
for (initialize; check_bool_exp; update) {
            //code;
}
```

2. **While loop: the syntax looks like this:**

```
while(boolean) {
          //code
}
```

***stops when Boolean becomes false***

3. **do-while loop: do-while loop is guaranteed to execute at least once.**

```
do {
         //code
} while (condition);
```

***Note the semicolon***

**Functions in java**

## We can use functions to separate logic like this:

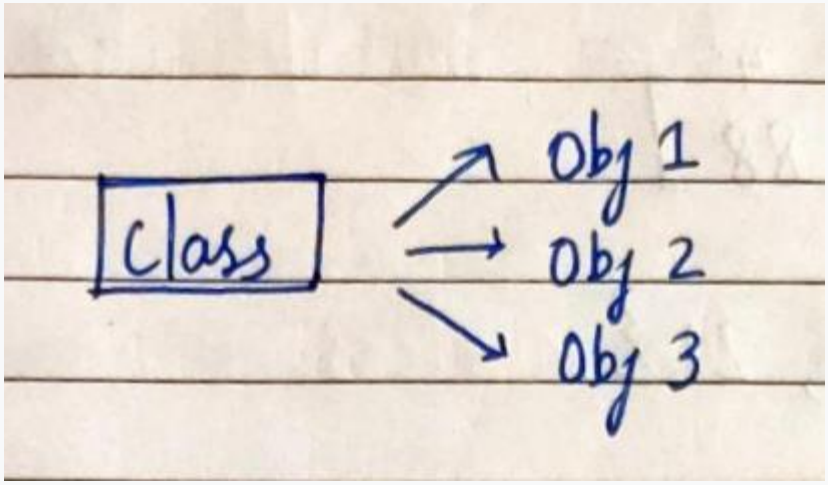
```
public static void harrysMethod (int a) {

                //code

            }
```

## This method can be called this:

```
harrysmethod(7);
```

### Object-oriented programming

## In oops, the class is a blueprint for creating objects.




### Syntax:

```
public class Harry {
    public static void thisMethod () {
        //code

    }
}
```

## The objects can be created like this:

```
Harry h = new Harry();
```

### Inheritance in java

## In java, we can create classes from another class like this:

```
public class Programmer extends Employee {
    private String language;
    // code
}
```

## Employee --> Programmer

### Arrays in java

## We can store a collection of similar items in an Array

```
[7, 10, 11, 21, 88]
```

```
int [] harry = { 1, 5, 9, 21};
System.out.println(harry[0]);
```

### Java collections framework

## The collections framework in java allows us to enjoy features like resalable arrays.

## An ArrayList is a class inside collections framework for creating resizable arrays.

## Array list 2 other collection methods, visit the java docs.

### Iterating through Array lists

## We can iterate through Array lists like this:

```
for (object o: harry) {   // Here harry is an Arraylist
    System.out.println ("object:"+ 0);
}
```

### How to view java Docs

## Java has very beautiful documentation where all the details from the API are listed.

**You can navigate to Google and search for the docs where you can further search for the java API you are interested in.**

## Chapter 2: Practice Set

**1 - Write a java program to print the multiplication table of a given number.**

**2 - Write a java program to add two numbers.**

**3 - Write a java program to find out the day of the week given the number.**

**4 – Write a java program to calculate income tax on a given income considering Indian laws.**

**5 – Write a java program to sum first n even numbers using for loops.**

**6 – Write a program to print the following pattern:**


## Chapter 3: Activities & Layouts

### What is an Activity?

**A single, focused thing that the user can do.**

**In layman's terms – Activity = Screen**

**The activity has layouts. //Layouts define how views are displayed.**

### Types of Layouts in Android

**If you go to the palette, you will find a lot of layouts. We can use any of these (LinearLayout, ConstraintLayout, etc.)**

#### Linear Layout

**Arranges its collection of views in a straight horizontal or vertical row.**

#### Required attributes of a LinearLayout

- **Layout width // Can be match-parent/ wrap-content/some value dp**
- **Layout height**
- **Orientation // Can be horizontal or vertical**

**Note: Never close your Emulator on every run when you change the app Emulator is a resource-hungry program & will take time to launch.**

#### Relative Layout

- **Elements are aligned relative to each other**
- **Attributes like android:layout_alignParentTop are used to declare positions of views.**

#### Legacy tab in Palette

**Some View Layouts in Android Studio are replaced with better counterparts.**

- **ListView is replaced with RecyclerView**
- **GridView is replaced with ConstraintLayout**
- **TabHost is replaced with TabLayout**
- **RelativeLayout is replaced with ConstraintLayout**

#### Important Notes

- **The component view shows certain warnings sometimes. Correcting these by following good practices is recommended.**
- **You can resize the preview as per the device of your liking in the Design view itself.**
- **Android Studio uses Gradle as the build system.**
- **The project structure of Android is very simple & straightforward.**
- **You don't have to remember all the attributes of every View**

#### Strings.xml file

**This file contains string resources in XML format**

```
<string name = 'app'> Harry </string>
```

In the above line of code, the app is identified and Harry is the value.

We can use  bold  and  Italic  tags inside the text in strings.xml. Other HTML tags are ignored Escape sequence characters like '\n' are also allowed.

### ScrollView

ScrollView is a ViewGroup used to create Scrollable Views. It contains just one child which can be greater than the screen height and can be scrolled.

Scrollview offers many attributes which is used to customize it.

### Horizontal ScrollView

It is very similar to ScrollView but instead of vertical scrolling, it provides functionality for horizontal scrolling.

### Logging in Android

We can log a message to the console using:

```
Log.d("Tag", "Log this message");
```

Similarly, we can use Log.i for info, Log.e for error, etc.

### Constraint Layout

Constraint Layout allows us to position widgets by applying constraints

We can add a Vaseline constraint on widgets inside the constraint layout by right-clicking > show baseline & finally adding the baseline constraints.

## Chapter 3: Practice Set

1. Create a layout that looks like this:
2. Create a layout that looks like this:
3. Create a layout that shows a bulk text scrollable vertically:
4. Create a layout that looks like a contact us page of a company.

# Chapter 4: Multiscreen Apps

## An Activity = Screen

**One activity in an app is specified as the "main" activity which is shown to the user when the app is launched.**

### Intent

**Whenever a new activity starts, the previous activity is stopped but the system preserves the activity in a stack.**

**This way when a new activity starts, that new activity is pushed onto the back stack and takes user focus.**

**An activity is started with an Intent. An intent is a message from one activity to another activity. We can pass information from one activity to another using Intent.**

### Types of Intents

**An Intent can be of two types:**

1. **Implicit Intent: The target component is not known and we have a general action to perform.**
2. **Explicit Intent: The target of the intent is known and a fully classified class name of a specific activity is also known.**

### Our first Multiscreen App

**We will now create an app that has an EditText on the first activity. The user enters a message and clicks a button to send this message to the second activity.**

**We will now create this App using the following steps:**

1. **Create a new app**
2. **Design the layout for the first app**
3. **Create button click listeners**
4. **Create the second activity**
5. **Add proper links and metadata in manifest.xml**
6. **Add an intent using the following code:**

```
Intent intent = new Intent(this, SecondActivity.class);
startActivity(intent)
```

### Sending data cross activities

**We can send data cross activities using intent extras.**

**Intent extras are key/value pairs in a Bundle.**

**A Bundle is a collection of data stored as key/value pairs.**

**The syntax for Intent putExtra looks like this:**

```
intent.putExtra("key","value");
```

**We can get this message from another activity using:**

```
Intent intent = getIntent();
String message = intent.getStringExtra("key");
```

### Implicit Intents

**In implicit intent, we initiate activity without knowing which app or activity will handle the task.**

**Ex: Take a photo, open this URL, etc.**

**Activities with matching intent filters opt in to perform the action.**

### Creating an App with Implicit Intent

**The following code starts an activity to open a URL**

```
String url = "Some URL" // (e.g. https://codewithharry.com)
Intent mint = newIntent(Intent.ACTION_VIEW, webpage);
if (intent.resolveActivity(getPackageManager())!=null){
        startActivity(intent);
}
else {
        //Cannot handle intent
}
```

**Similarly, we can handle intents with other actions.**

**For e.g. open a location, share text, etc.**

See android docs for more.

## Chapter 4: Practice Set

1. Create a layout in Android which looks like this
2. Create an android app capable of opening your favorite websites with a single click
3. Create a quiz app using the layout in Question.1
4. Create an App capable of sending Email to the user. The app should take Email, subject, and message as input.

## Chapter 5: ListViews & RecyclerView

In Android, a scrollable list of items is implemented using a ListView. The data is populated into the list using an Adapter.

Adapter converts an Array/Arraylist in to view items.

### Array Adapter

Array Adapter is used to display a set of items in a ListView.

```
ArrayAdapter <String> ad=new ArrayAdapter(this, R.layout.list_item,SArr)
listView.SetAdapter(ad);
```

This sets the content of SArr to list view

### Custom Array Adapter

We can create custom Array Adapters by creating a model and a class (e.g. MyAdapter) which extends ArrayAdapter. We can pass our string array to MyAdapter like this:

```
MyAdapter ad = new MyAdapter(this, SArr);
listView.SetAdapter(ad)
```

### Why use ListViews

It will be very hectic to create a scrollable view where data is coming from a Data Source. Just imagine how hectic would it be to otherwise populate data into a view to show it to the user.

**Hence Listviews are used due to the following reasons:**

- **User & Developer Friendly**
- **Optimized to some extend**
- **Easy to customize**
- **Simple Lists can be created in few lines of code**

**Adding onClick listener to items**

**We can override onItemClick method to add click listeners as shown below:**

```
listview.setOnItemClickListener(new OnItemClickListener(){

        ...

}
```

**Built-in XML layouts**

**Android provides you a list of built-in layouts to be used within list views.**

**E.g. android.R.layout.Simple_list_item_1**

**android.R.layout.Simple_list_item_2**

**... etc.**

**RecyclerView**

**It's simply a better version of listViews**

**In ListViews, memory is directly proportional to the number of items**

**RecyclerView solves this problem by recycling the existing views hence saving up on memory**

**We simply update the Adapter in RecylerView to an Adapter that is capable of Recycling the Views**

**ListViews Vs RecyclerViews**

**ListViews:**

- **The user keeps scrolling which adds Views and hence more memory is consumed.**

### RecyclerViews:

- **Views are recycled when a user scrolls resulting in a performance boost.**

### How to implement a RecyclerView?

**A RecyclerView can be implemented just like ListViews using an Adapter. All the major changes are done to the Adapter. Android Docs has a page that can be used as a starter template for implementing RecyclerViews.**

## Chapter 5: Practice Set

1. **Create an app using ListViews to display the contents of a string array with onClick listeners.**
2. **Create an app using Recyclerview from Question-1**
3. **Create an app using REcyclerViews to display contents of the "Contact" array which looks like this:**

```
public class contact {

        private int Sno;
        private String phoneNo;
        private String name;
}
```

## Chapter 6: Media Player

**Media is a crucial component of Android**

**Media = Audio + Video + Images**

### The MediaPlayer class

**This class provides APIs for playing a variety of media types. We can simply create an instance, add music and play it.**

### Adding music to the android "raw" directory

**We can add an mp3 file to our res/raw folder. If the raw folder is not present, you need to create one.**

### Playing our first music

We can create an instance of MediaPlayer and play our first music by adding the following lines inside onCreate

```
mediaPlayer = MediaPlayer.create(this, R.raw.music); // plays music.mp3
mediaPlayer.start()
```

Similarly, we can use mediaPlayer.pause() to pause the music from the MediaPlayer

### Playing music from the web

We can play music from the web using the setDataSource method of media player.

The following steps play music from the web

Step 1: Create a new MediaPlayer instance

```
mediaPlayer = new MediaPlayer();
```

Step 2: Set the data source

```
mediaPlayer.setDataSource("https://audio.source.wm") //use mp3 URL
```

Step 3: Add android.permission.INTERNET to manifest

Step 4: Add android:usesCleartextTraffic = 'true' in the applications tag of the manifest (Android Manifest.xml)

Step 5: Add Onprepared listener to the MediaPlayer and override required methods.

Step 6: Run mediaPlayer.prepareAsync(); method to start preparing the MediaPlayer.

### Adding SeekBar

Android SDK provides a SeekBar widget which allows developers to add a progress bar for media Execution.

Further, we can add OnSeekBarChangeListener() to take action when the seekbar is changed.

This can be done like this:

```
SeekBar.setMax(mediaPlayer.getDuration());
```

```
SeekBar.setOnSeekBarChangeListener(new SeekBar.OnSeek...        //Use autocomplete and override methods
```

### Playing Videos

We can use the same MediaPlayer class to play videos on Android.

In order to display videos we use a SurfaceView like this:

```
mediaPlayer = MediaPlayer.create(this,R.raw.vid)

SurfaceHolder h = SurfaceView.getHolder();

h.addCallback(new SurfaceHolder.Callback(){

        ... // override methods here

        });
```

## Chapter 6: Practice Set

1. Create an Android App which plays your favorite song on loop
2. Add a seekbar in Question-1 for you to seek a specific part in your favorite song
3. Create an Android app that plays three favorite files of a user from the webserver
4. Create an App that plays two videos simultaneously on one Android Screen.
5. Stop one when another is played from your app in Question-4.

## Chapter 7: Storing Data

Saving data makes apps livelier

Data persistence = Storing & Saving the data

In android, we can save data using these common ways:

- Text file – (Not Efficient)
- Shared Preferences – (Somewhat Efficient)
- Database – (Highly Efficient)

### Shared Preferences

**Shared Preferences allow us to save and retrieve data in the form of a key, value pair**

**We can access Shared preferences like this:**

```
SharedPreferences sP = getSharedPreferences(MESSAGE, MODE_PRIVATE)
SharedPreferences.Editor ed = sP.edit();
ed.putString("text", "This is me");  //Storing Data
ed.apply();
sP.getString("text", "default value"); //Retrieve Data
```

### Working with Database

**We can use SQLite database to work with databases in Android.**

**We can create a class extending from SQLiteOpenHelper and use it for CRUD operations.**

## Chapter 7: Practice Set

1. **Store the HiScore of a game using SharedPreferences and fetch it into a TextView**
2. **Create a database to store data of Employees using SQLiteOpenHelper class**
3. **Add an Entry to the database created in Question-2**
4. **Fetch an Entry from the database created in Question-2**