**Android Development**

(Beginner to Advanced)

# What is Android?

License free

* An Android is an **Open**-**source**, **Linux** based operating system. It is developed by the Open Handset Alliance, led by Google and other companies. Google has further developed Android TV for televisions, Android auto cars and wear OS for wrist watches.
* **What is Operating System?**
* An Operating System is a **software** program that enables the **Hardware** to communicate and operate with the software program. An OS is an User Interface between User and Hardware. It provides a software platform on top of which other programs called application programs can run.
* **What is Linux?**
* Linux is a family of free and open-source software operating systems built around the Linux kernel. The development of Linux is one of the most prominent examples of free and opensource software application.
* **What is Open-Source?**
* The source code of a computer program is made available **free of charge** to the general public. Users can view the code that comprises the software and make **any kind of changes** to it they want.
* Anyone can take the source code and distribute their **own program** from it.



* **History of Android:**

The history and versions of android are interesting to know. Initially, Andy Rubin founded Android Incorporation in Palo Alto, California, United States in October, 2003.

* In 17th August 2005, Google acquired android Incorporation. Since then, it is in the subsidiary of Google Incorporation.
* The key employees of Android Incorporation are Andy Rubin, Rich Miner, Chris White and Nick Sears.
* Originally intended for camera but shifted to smart phones later because of low market for camera only.
* In 2007, Google announces the development of android OS.
* In 2008, HTC launched the first android mobile.

(Up to 2023 these are versions released)

|  |  |  |
| --- | --- | --- |
| **Android Version** | **Name** | **Year** |
| 1.0 | Base 0 | 23 Sep,2008 |
| 1.1 | Base 1\_1 | --------------- |
| 1.5 | Cup Cake | 27 Apr,2009 |
| 1.6 | Donut | 15 Sep,2009 |
| 2.0 | Eclair | 26 Oct,2009 |
| 2.0.1 | Éclair 0\_1 | 26 Oct,2009 |
| 2.1.X | Éclair MR1 | --------------- |
| 2.2.X | Froyo | 20 May,2010 |
| 2.3 | Ginger Bread | 6 Dec,2010 |
| 2.3.4 | Ginger Bread MR1 | --------------- |
| 3.0.X | Honeycomb | 22 Feb,2011 |
| 3.1.X | Honeycomb MR1 | --------------- |
| 3.2 | Honeycomb MR2 | --------------- |
| 4.0 | Ice Cream Sandwich | 18 Oct,2011 |
| 4.1 | Jelly Bean | 9 Jul,2012 |
| 4.4 | Kitkat | 31 Oct /3 Sep,2013 |
| 4.4.W | Kitkat Watch | --------------- |
| 5.0 | Lollipop | 12 Nov,2014 |
| 6.0 | Marshmallow | 5 Oct,2015 |
| 7.0 | Nougat | 22 Aug,2016 |
| 8.0 | Oreo | 21 Aug,2017 |
| 9 | Pie | 6 Aug,2018 |
| 10 | Android 10 | 3 Sep,2019 |
| 11 | Android 11 | 8 Sep.2020 |
| 12 | Android 12 | 4 Oct, 2021 |
| 13 | Android 13 | 15 Aug,2022 |
| 14 | Android 14(Upcoming) | -- Aug, 2023 |

* **What is IDE in Android?**

An integrated development environment (IDE) is a software application that helps programmers develop software code efficiently. It increases developer productivity by combining capabilities such as software editing, building, testing, and packaging in an easy-to-use application. Just as writers use text editors and accountants use spreadsheets, software developers use IDE’s to make their job easier. To perform coding we use **Android Studio** for developing the code.

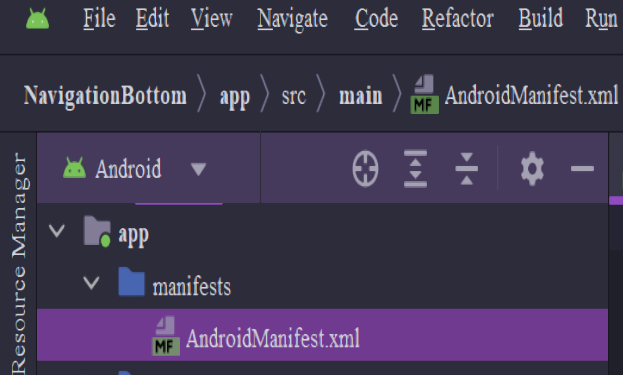
* latest version of Android Studio ([Click here to Download](https://developer.android.com/studio)).
* latest version of Java ([Click here to Download](https://www.oracle.com/in/java/technologies/downloads/#jdk20-windows)).

**Main feature of IDE’s:**

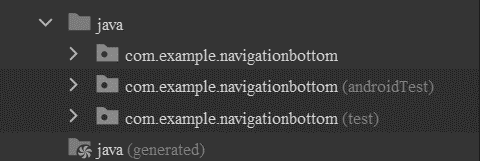
* + Editing source code
  + Syntax highlighting
  + Auto Complete (Suggestions)
  + Building Executables
  + Debugging (Manual Testing)
* **Front End Language:**
* Its all about what a User able to see. It’s may be buttons, images, layout, input-field, toggle/radio buttons etc.
* To design these components in android we need XML and Android programming ([Java](Java%20Course.docx)/Kotlin).
* **Back End Language:**
* Backend of anything is not visible to general user or administrator. It can be a piece of code or a program running on the server machine to serve the user need.
* Database is also a part of backend (i.e., SQLite, NoSQL, Firebase, MongoDB etc.)
* **What is Apk?**
* The extension of Apk file is .apk. It is installed on Android Operating System.
* It consists of:
* Application code (.dex files)
* Manifest file
* Assets
* Resource files
* **What is Android Studio exactly means?**

An android project 4 types of files they are mainly manifestfile.xml, java, res and Gradle Scripts. While creating the project there are 2 files will be created (activity\_main.xml, MainActivity.java)

(NOTE: while creating any folder don’t use –(hypen)/any number (0,1 etc)/any specialcharacter (#,$ etc)



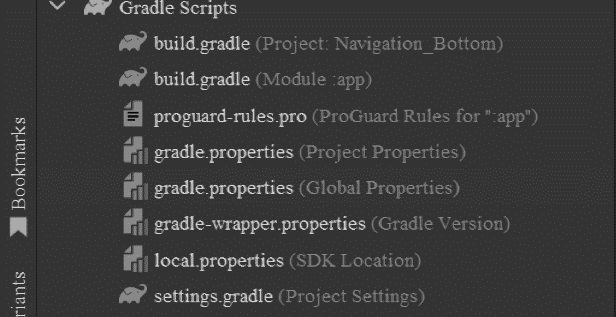
It provides essential information about our android app to the Android OS. It must be present in the root directory of your app’s project. It contains various elements and attributes like Package Name, Permissions (use of Internet etc.), Activities, Intent Filters, App Icon, Themes, Versions Requirements etc.



It plays a significant role in developing android applications. Java is the primary language used to write the logic and functionality of about Android apps. In contains Activity class, Java packages, Event handling, XML layouts, Java methods, Resource access, Libraries and API’s, Testing etc.



It is UI design which contains XML files, images, strings, colors, layouts, and many more. Some of the sub-directories are drawable, layout, values, mipmap, anim, menu, raw, xml. In this you can add different types of image formats like JPG, JPEG, PNG, GIF, WebP, SVG while developing the android you can use these of images and you can add the layouts.



“Gradle” is an important directory that contains Gradle-related files. Gradle is the build system used in the android projects to automate the build process, manage dependencies and other tasks.It is a status bar. It contain gradle-wrapper properties,gradlew and gradle.bat,build.gradle(Project),build.gradle(Module),gradle.properties and settings.gradle.

* **What is Android Virtual Device (AVD)?**

Virtual Device is a device which is created virtually(imaginary) and it is used for testing and debugging the applications without the need of Physical device (Real device) on our hand. Virtual devices allow the developer to emulate the various configurations, screen-sizes which you can customize to your need, hardware features and different versions of API’s are available in android. The other name of Virtual device is Emulator.

* There are 4 stages on running the AVD:
* Gradle Building: It creates .apk file
* Indexing: It will create the index of building .apk file
* Installing: It is ready to launch mode
* Launch mode: It shows the activity to the user
* **Debugging in Android:** In simple words, Debugging is nothing remove the error/finding and fixing mistakes or problems in your app. It’s like being like a detective for the app, where your look for clues (error messages, unexpected behavior)
* **How to fix it?** We can fix the error through log class
* **Log:** Log is class and used to display system messages, such as when a [garbage collection](https://www.google.com/search?q=garbAge+collection+in+java&oq=garbAge+collection+in+java+&aqs=chrome..69i57.11163j0j1&sourceid=chrome&ie=UTF-8) occurs, and messages that you added to your app with log class.

It displays messages in real time and keeps a history so you can view older messages.

* Methods of log class:
* **Log.v**(String tag, String message): Used for **verbose**-level log messages. These messages provide detailed information that can help with debugging.
* **Log.d**(String tag, String message): Used for **debug**-level log messages. These messages are typically used to track the flow of your code during development and debugging.
* **Log.i**(String tag, String message): Used for **informational** log messages. These messages provide useful information about the execution of your app.
* **Log.w**(String tag, String message): Used for **warning** log messages. These messages indicate potential issues or situations that could cause problems.
* **Log.e**(String tag, String message): Used for **error** log message. These messages indicate errors or exceptions that have occurred on the app.
* **Logcat:**
* Same as log, just difference is that Logcat always check the LOG’s property that is true or false before logging any message.
* It can be set to true in the debugging build type and to false in the release type.
* **Toast:** Toast is a class which is used to display information for the short period of time. A toast contains message to displayed quickly and disappears after sometime.

Message

Syntax:

|  |
| --- |
| Toast toast = Toast.makeText (getApplicationContext(),"Hello Javatpoint", Toast.LENGTH\_SHORT); |

* **What is XML?**
* XML stands E**X**tensible **M**arkup **L**anguage.
* XML is used for storing and transferring **data**.
* XML is a markup language much like **HTML**.
* Unlike HTML, XML is **case-sensitive**, requires each tag is closed properly, and preserves whitespace.
* XML tags need not to be **predefined** like HTML(therefore known as extensible)
* A markup language is a language that **annotates** text so that the computer can **manipulate** that text.
* Most markup languages are **human-readable** because annotations are written in a way to distinguish them from text itself.
* It was designed to be **self-descriptive**.
* **XML in Android:**
* We create XML **layouts** in android, and later alter them using Java logic.
* **Resources** are the additional files and static content an application needs, such as animations, color schemes, layouts, menu layouts.
* Each layout file, must contain one (and only one!) **root** element(tag).
* Here layout is nothing it a combinations of **multiple-views** (View, View Group & Root View)
* **Views:**
* It represents a rectangular are of the screen, and is responsible **for displaying information or content**, and event handling.
* Text, images, buttons, radio button, checkbox etc. are the view in android.
* **View Group:**
* It is essentially an **“invisible container”** that holds multiple Views or View Groups together, and defines their layout properties.
* Common View Groups
* A **List View** displays a list of scrollable items.
* A **Grid View** displays items in a two-dimensional scrollable grid.
* A **Table Layout** groups views into rows and columns.
* **Root View:**
* It is a Root Element of XML layout file.
* Common View Groups
* A **Linear Layout** aligns its contents into a single direction, whether vertical or horizontal.
* A **Relative Layout** displays its child content in positions relative to the parent.
* A **Frame Layout** is a placeholder on a screen that displays only a single view (Fragments).

Root View

<?xml version="1.0" encoding="utf-8"?>  
<RelativeLayout  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"   
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 tools:context=".MainActivity">  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:text="Hello World!" />

Attributess

View (Child)

</RelativeLayout>

* **Layouts in Android:** Layouts in Android are a key concept for designing and arranging UI components within an app’s screen. They define the visual structure and positioning of UI elements like buttons, text views, images, plaintext, image button etc. There are several layouts in android. The layouts are:
* Linear Layout: It is used to arrange the **attributes vertically or horizontally** in a form side by side (or) a sequence pattern.
* Relative Layout: It is a layout which is used for **overlapping** (view on view).
* Constraint Layout:
* Frame Layout:
* Grid Layout:
* Table Layout:
* Scroll View:
* View Page:
* Card View:
* **Views in Android:**
* xlmns:(namespace URI):
* URI stands for Uniform Resource Indicator.
* In computing, it is a string of computing used to identify a name of a resource.
* Such identification enables interaction with representations of the resource over a network, typically the World Wide Web, using specific protocols.
* Eg: [xmlns:app="http://schemas.android.com/apk/res-auto"](xmlns:app=%22http://schemas.android.com/apk/res-auto%22)

Important attributes:

* android:layout\_width: “match\_parent”
* android:layout\_height: “match\_parent”

# **What is need of Java in Android?**

* A general-purpose programming Language made for developers to **write once run anywhere** that is compiled Java code can run on all platforms that support Java. Java is also known as **platform-independent**.
* It is used for:
* Mobile Applications (specially android apps)
* Desktop applications
* Web applications
* Web servers and application servers
* Games
* Database connection etc.

## What is JVM?

* Writing a program is done by java programmer(you).
* The compilation is done by JAVAC compiler which is a primary Java compiler included in the Java Development Kit (JDK). It takes java program as input and generates **bytecode** as output.
* It is saved as .**class** file by the compiler.
* In running phase of a program, **JVM** executes the bytecode generated by the compiler.

## What is JDK?

* It is a complete java development kit that includes everything including compiler, Java Runtime Environment (JRE), Java debuggers, java docs etc.
* For the program to execute in java, we need to install JDK in our computer in order to create, compile and run the java program.

## What is JRE?

* JDK includes JRE and JVM.
* JRE installation on our computers allow the java program to run.
* For running the java program, a computer needs a JRE.

# **About Java**

* What is Package in Java? A package in java s used to group/collection of related classes. Think of it as a folder in a file directory. We use packages to avoid name conflicts, and to write a better maintainable code.
* Packages are two types Built in Packages & User-defined package.
* What is Class? A class is an entity that determines how an object will behave and what the object will contain. It contains variables(fields), methods, objects & blocks.
* Data types: It is a special keyword used to allocate sufficient memory space for the data (or) It is used for representing the data in main memory (RAM).
* There are two types Primitive (Boolean, char, int, short, long, float & double) and Non-primitive type (String, Class, Interface & Arrays).
* Variables:It is a container for storing data values. These are assigned with a data type. It is a name of memory location.
* Methods:It is a collection of statements that are grouped together to perform an operation. It takes some parameters, performs some computations, and then optionally returns a value (or object). It is sued to remove redundant(repetitive) code and to reuse code.
* Encapsulation: It is a mechanism of wrapping the date(variables) and code acting on the data(methods) together as a single unit. It is used to hide the data, through access modifiers (public, private, protected and default). You want to access the values use getters and setters.

(Here is an example for encapsulation)

Customerid: 1

CustomerName: Kiran

CustomerSalary :28666.66

|  |
| --- |
| **package** com.example.Encapsulation;  **class** Customerdetails{  **private** **int** id;  **private** String Name;  **private** **double** salary;  **public** **int** getId() {  **return** id;  }  **public** **void** setId(**int** id) {  **this**.id = id;  }  **public** String getName() {  **return** Name;  }  **public** **void** setName(String name) {  Name = name;  }  **public** **double** getSalary() {  **return** salary;  }  **public** **void** setSalary(**double** salary) {  **this**.salary = salary;  }}  **public** **class** Owner {  **public** **static** **void** main(String[] args) {  Customerdetails cd = **new** Customerdetails();  cd.setId(1);  cd.setName("Kiran");  cd.setSalary(28666.66);  System.***out***.println("Customerid: "+ cd.getId());  System.***out***.println("CustomerName: "+cd.getName());  System.***out***.println("CustomerSalary :"+cd.getSalary());  }} |

* Keywords**:** Keywords are the words in a language that are used for some internal process pr represent some predefined actions.
* These words are therefore not allowed to use as a variable names or objects.
* Doing this will result into a compile time error.
* Therefore, also known as **Reserved** words.
* Identifiers**:** Identifiers in Java are symbolic names used for identification.
* They can be a class name, variable name, method name, package name, constant name, and more.
* Keywords/Reserve words cannot be used as identifiers.
* Identifiers are **case-sensitive**.
* Method Overloading**:** If a class has multiple methods having **same class**, **same name** but **different no of parameters**, it is called Method Overloading.
* It increases the readability of the program.
* It is a **Compile time** Polymorphism(static).

3

12

Eg:

|  |
| --- |
| **package** com.example.Polymorphism;  **public** **class** Overloading{    **public** **static** **int** add(**int** a,**int** b) {  **return** a+b;  }  **public** **static** **int** add(**int** a,**int** b,**int** c) {  **return** a+b+c;  }  **public** **static** **void** main(String[] args) {    System.***out***.println(*add*(1,2 ));  System.***out***.println(*add*(3,4,5 ));  }    } |

* Method Overriding:If subclass (child class) has the same method as declared in the parent class, it is known as method overriding.
* When a method in a sub-class has **the same name, same parameters** & **same return type** as a method in its super-class, then the method in the sub-class is said to be **override** the method in the super-class.
* It is called as **Run time** Polymorphism (Dynamic).

15

7.699999999999999

Eg:

|  |
| --- |
| **package** com.example.Polymorphism;  **public** **class** Overriding {  **public** **int** add(**int** i, **int** j) {  **return** i+j;  }  **public** **double** add(**double** x, **double** y) {  **return** x+y;  }  **public** **static** **void** main(String[] args) {  Overriding ov1 = **new** Overriding();  System.***out***.println(ov1.add(5,10));  System.***out***.println(ov1.add(3.4,4.3));  }  } |

* Inheritance:It is the mechanism in java by which one class is allow to inherit the features (fields and methods) of another class.
* One object acquires all the properties and behaviors of a parent object.
* It is an important part of Oops (Object Oriented Programing System).
* **Sub-Class** can reuse methods and fields of the parameters of **Super-Class**.
* It supports Single Inheritance, Multi-level inheritance, Hierrachial Inheritance but it doesn’t support Multiple inheritance due to ambiguity(confusion)🙄.
* Constructors:Method with same name as class name without return type. Constructors are of two types Default constructor and Parametrized constructor.
* Interface:An interface is a reference type in Java.
* It is similar to class
* It is a collection of abstract methods.
* A class implements an interface, thereby inheriting the abstract methods of the interface.
* Only method signature, no body.
* Interfaces specify what a class must do and not how. It is a blueprint of the class

## this, static & final keywords:

* **this** is a refence variable that refers to the current object. It is a keyword in Java language represents **current class** object. This keyword can be used to refer to any member of the current object from within an instance method or a constructor.
* this can be used to refer current class instance variable.
* this can be used to invoke current class method (Implicitly).
* this () can be used to invoke current class constructor.
* this can be passes as an argument in the method call.
* this can be passed as argument in the constructor call.
* this can be used to return the current class instance from the method.
* **static** keyword indicates that the particular member belongs to a type itself, rather than to an instance of the type. This means that only one instance of that static member is created which is shared across all instances of the class.
* **final** keyword can be used while declaring an entity. Using the final keyword means that the value can’t be modified in the future.
* Conditional Statements**:** These are the statements which are executed on based on the condition whether it returns true or false like a Boolean value.
* Use **if** to specify a block of code to be executed, if a specified condition is true.
* Use **else** to specify a block of code to executed, if the same condition is false.
* Use **else if** to specify a new condition to test, if the first condition is false.
* Use **switch** to specify many alternative blocks of code to be executed.
* Loops:Loops are used to repeat a block of code. Java contains 4 loops. They are for loop, for each, while loop, do while loop.
* Java **while** loopis used to run a specific code until a certain condition is met.
* **do while** loop is similar to while, however while loop is executed once before the test expression is checked**.**
* Strings**:** sequence of characters/ group of characters/ collection of characters.

**Design in Android (Layouts):**

* **URI:** The most common URI is the Uniform Resource Locator (URI) which identifies an internet domain address. Another, not so common type of URI is the Uniform Resource Name (URN).

**XML Namespace:** XML namespaces are used for providing **uniquely named** elements and attributes in an XML document.

xmlns:android describes the android namespace.

|  |
| --- |
| Eg: xmlns:android="http://schemas.android.com/apk/res/android" |

* This URI provides it will be **uniquely identify** the attributes what will you design

|  |
| --- |
| [xmlns:app="http://schemas.android.com/apk/res-auto"](xmlns:app=%22http://schemas.android.com/apk/res-auto%22) |

* This is used to identify the View, View Group and to use **Third party libraries**. It is optional

|  |
| --- |
| xmlns:tools="http://schemas.android.com/tools" |

* This is used to identify the **XML namespace** it is generally used in Layout

Also suppose we write our textview widget with different features compared to android textview, android namespace helps to distinguish between our customer textview widget and android textview widget.

* **UI Design:** UI design in android focusses on creating visually appealing and user-friendly interfaces for mobile apps. Here are some basic views in Android they are:

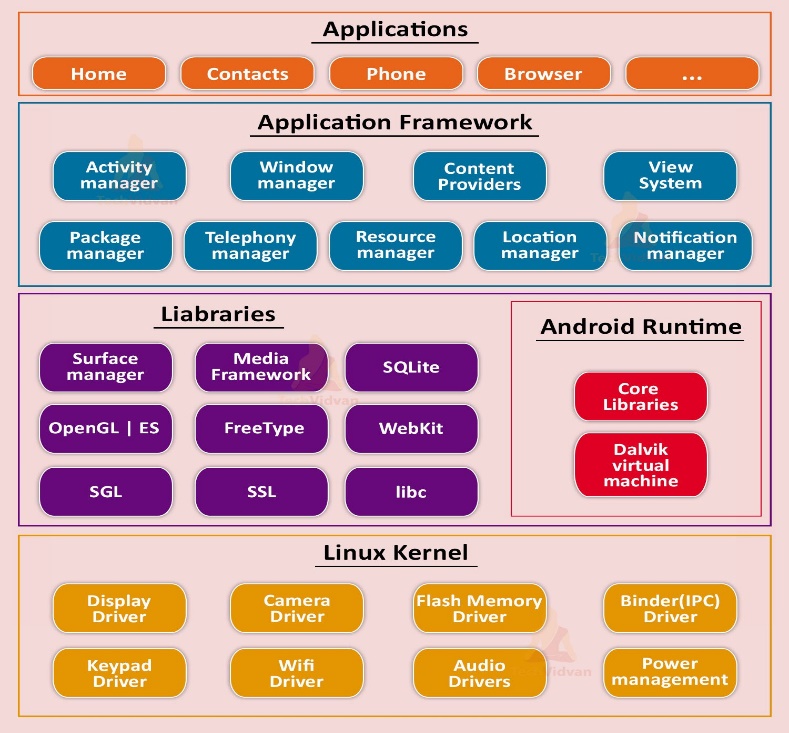
Button: It is used to interact with app and perform the actions like Submit, Play, Send.

Text View: It is used to display text on the screen.

Image View: It is used to display the image on the screen in formats (jpg, jpeg, svg etc)

Edit Text: It displays the input of the user.

* **Android Architecture:**



It provides functionalities such as power management,process management and memory management. It provides a level of abstraction between device hardware and upper layers of Android software stack.

* **Linux kernel** is a heart of android OS and present at the bottom of android architecture and it contains essential hardware drivers like display, audio, camera, bluetooth, wifi etc and it is a **root layer**.
* It provides functionalities such as power management, process management and memory management and it provides a level of abstraction between device hardware & upper layers of android software stack.
* **Libraries** can help in simplifying development, improviing efficiency and maintaining a clean codebase. It provides native libraries such as:
* SQLite: It provides various classes used for database management.
* WebKit: It is a web browser engine used to display internet content.
* SSL: It provides internet security.
* OpenGL: It is a Java interface to the OpenGL ES 3D graphics rendering API.
* **Android Runtime:** It is third section which is also present on the second layer from bottom of Android architecture. It provdies cooore libraries and Dalvik Virtual Machine(DVM).
* These core libraries enable android developers android applications using standard Java programming language. DVM is a major component of android OS. It is same as like Java Virtual Machine(JVM).
* DVM is responsible for running android applications. It consumes less memory and provides very fast perfromance.
* **Application Framework:** It provides various API’s like activity manager, resorce manager, content providers, telephony manager etc.
* It is present at the top. Various applications created by developers like games, contacts, browser etc are installed on this layer. In this it contain of 3 main types:
* Native apps
* Third Party apps
* Developer apps

(Overall, the Android Architecture promotes modularity, allowing developers to coroeate apps that leverage existing components and libraries while maintaining seperation between different aspects of the system. This architecture has evolved over the years with updates to Android versions, providing developers with better tools and feature for building innovative and efficent applications)

* **Android Components:** There are 4 components
* Activities: An activity represents a single screen with a UI, in-short activity performs actions on the screen.

|  |
| --- |
| public class MainActivity extends Activity{  } |

* Services: A service is a component that runs in the background to perfrom long-running operations.

|  |
| --- |
| public class MyService extends Service{  } |

* Broadcast Receivers: It is simply respond to broadcast messages from other applications or from the system.

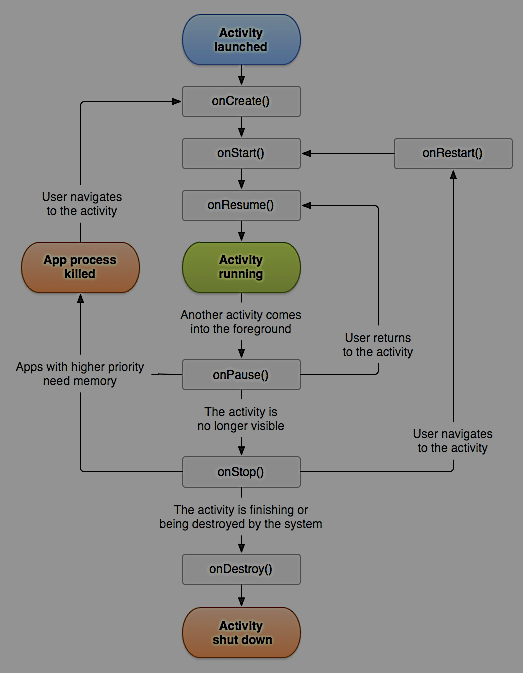
|  |
| --- |
| public class MyReceiver extends BroadcastReceiver{  public void onReceive(content,intent){  }} |

* Content Providers: A content provider component supplies data from one application to others on request. Such requetrs are handled ny the methods of the **ContentResolver** class.

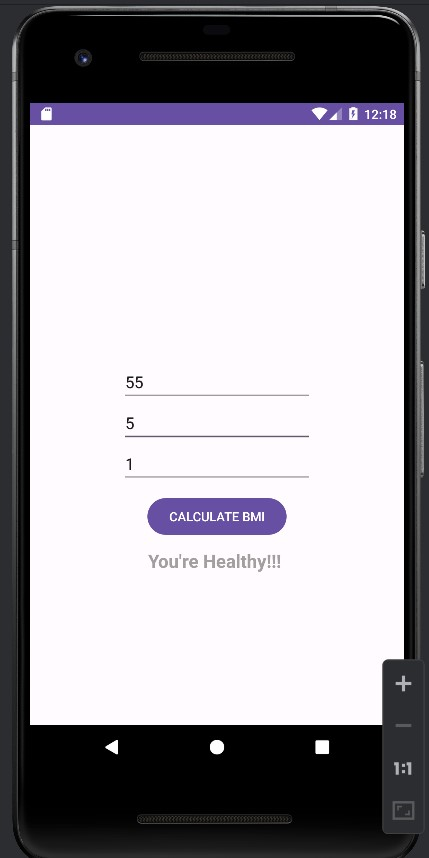
|  |
| --- |
| public class MyContentProvider extends ContentProvider{  public void onCreate(){  }  } |

* **Activity Lifecycle:** There are 7 seven stages in lifecycle**.**

|  |  |
| --- | --- |
| **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. |
| onPause() | called when activity is not visible to the user. |
| onStop() | called when activity is no longer visible to the user. |
| onRestart() | called after your activity is stopped, prior to start. |
| onDestroy() | called before the activity is destroyed. |



* **Creating First Android App of BMI:**



MainActivity.xml

|  |
| --- |
| <?xml version="1.0" encoding="utf-8"?>  <LinearLayout  xmlns:android="http://schemas.android.com/apk/res/android"  xmlns:tools="http://schemas.android.com/tools"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:orientation="vertical"  android:gravity="center"  android:layout\_marginTop="260dp"  tools:context=".MainActivity">  <EditText  android:id="@+id/Weight"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:ems="10"  android:hint="Enter Your Weight"  android:inputType="number"/>  <EditText  android:id="@+id/HeightFt"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:ems="10"  android:hint="Enter Your Height(in Ft)"  android:inputType="number"/>  <EditText  android:id="@+id/HeightIn"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:ems="10"  android:hint="Enter Your Height(in Inc)"  android:inputType="number"/>  <Button  android:id="@+id/btn"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="CALCULATE BMI"  android:layout\_marginTop="11dp"/>  <TextView  android:id="@+id/txt"  android:layout\_width="wrap\_content"  android:layout\_height="wrap\_content"  android:text="Result"  android:textColor="#FCA39E9E"  android:textSize="20sp"  android:textStyle="bold"  android:layout\_marginTop="11dp"/>  </LinearLayout> |

package com.example.calcuatorbmi;

import androidx.appcompat.app.AppCompatActivity;  
import android.os.Bundle;  
import android.view.View;  
import android.widget.Button;  
import android.widget.EditText;  
import android.widget.TextView;  
  
public class MainActivity extends AppCompatActivity {  
  
 @Override  
 protected void onCreate(Bundle savedInstanceState) {  
 super.onCreate(savedInstanceState);  
 setContentView(R.layout.*activity\_main*);  
  
 EditText edtWeight, edtHeightFt, edtHeightIn;  
 Button btnCalcuate;  
 TextView txtResult;  
  
 edtWeight = findViewById(R.id.*Weight*);  
 edtHeightFt = findViewById(R.id.*HeightFt*);  
 edtHeightIn = findViewById(R.id.*HeightIn*);  
 btnCalcuate = findViewById(R.id.*btn*);  
 txtResult = findViewById(R.id.*txt*);  
  
 btnCalcuate.setOnClickListener(new View.OnClickListener(){  
 @Override  
 public void onClick(View v) {  
 int wt = Integer.*parseInt*(edtWeight.getText().toString());  
 int ft = Integer.*parseInt*(edtHeightFt.getText().toString());  
 int in = Integer.*parseInt*(edtHeightIn.getText().toString());  
  
 int totalIn = ft\*12 + in;  
 double totalCm = totalIn\*2.53;  
 double totalM = totalCm/100;  
 double bmi = wt/(totalM\*totalM);  
  
 if(bmi>25){  
 txtResult.setText("You're Overweight");  
 } else if (bmi<18) {  
 txtResult.setText( "You're Underweight");  
 }else {  
 txtResult.setText("You're Healthy!!!");  
 }  
 }  
 });  
}}

* **Making TIC-TAC-TOE app:**

**activity\_main.xml**

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout

xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:gravity="center"

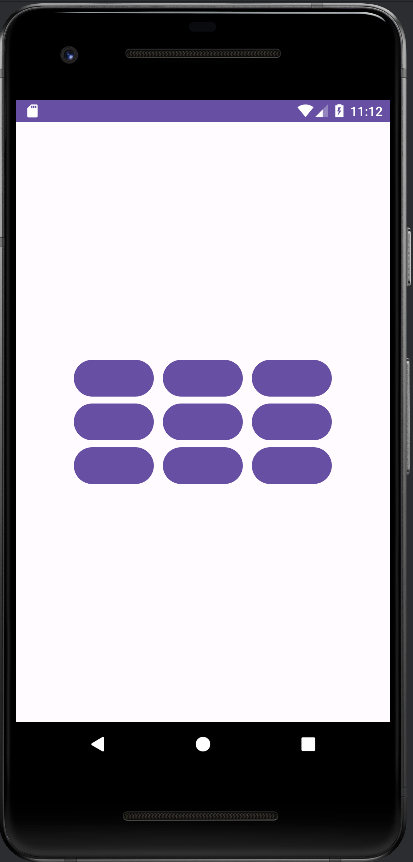
tools:context=".MainActivity">

<GridLayout

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:rowCount="3" UI:



android:columnCount="3">

<Button

android:id="@+id/btn1"

android:onClick="Check"/>

<Button

android:id="@+id/btn2"

android:layout\_marginLeft="10dp"

android:onClick="Check"/>

<Button

android:id="@+id/btn3"

android:layout\_marginLeft="10dp"

android:onClick="Check"/>

<Button

android:id="@+id/btn4"

android:onClick="Check"/>

<Button

android:id="@+id/btn5"

android:layout\_marginLeft="10dp"

android:onClick="Check" />

<Button

android:id="@+id/btn6"

android:layout\_marginLeft="10dp"

android:onClick="Check"/>

<Button

android:id="@+id/btn7"

android:onClick="Check"/>

<Button

android:id="@+id/btn8"

android:layout\_marginLeft="10dp"

android:onClick="Check"/>

<Button

android:id="@+id/btn9"

android:layout\_marginLeft="10dp"

android:onClick="Check"/>

</GridLayout>

</LinearLayout>

**MainActivity.java**

package com.example.tictaetoeprogram;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;

import android.view.View;

import android.widget.Button;

import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

Button bt1,bt2,bt3,bt4,bt5,bt6,bt7,bt8,bt9;

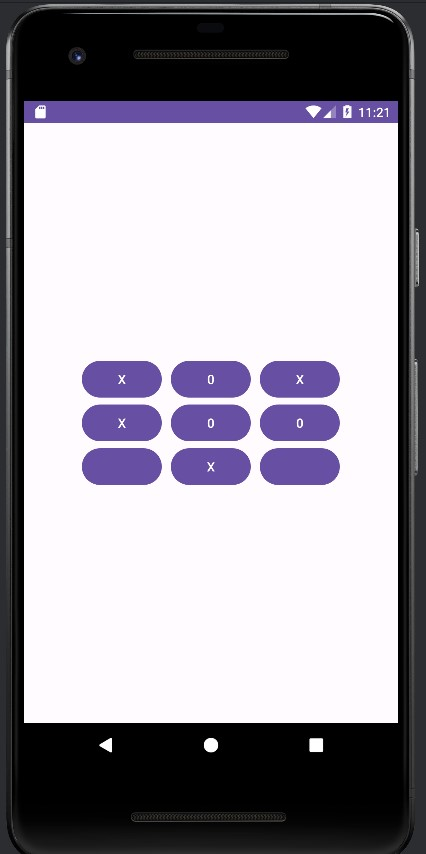
String b1,b2,b3,b4,b5,b6,b7,b8,b9;

int flag = 0;

int count = 0;

@Override

protected void onCreate(Bundle savedInstanceState) { Output:



super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

init();

}

public void init(){

bt1 = findViewById(R.id.btn1);

bt2 = findViewById(R.id.btn2);

bt3 = findViewById(R.id.btn3);

bt4 = findViewById(R.id.btn4);

bt5 = findViewById(R.id.btn5);

bt6 = findViewById(R.id.btn6);

bt7 = findViewById(R.id.btn7);

bt8 = findViewById(R.id.btn8);

bt9 = findViewById(R.id.btn9);

}

public void Check(View view){

Button btnCurrent = (Button) view;

if(btnCurrent.getText().toString().equals(" ")){ }

count++;

if(flag == 0){

btnCurrent.setText("X");

flag = 1;

}else{

btnCurrent.setText("0");

flag = 0;}

if(count>4){

b1 = bt1.getText().toString();

b2 = bt2.getText().toString();

b3 = bt3.getText().toString();

b4 = bt4.getText().toString();

b5= bt5.getText().toString();

b6 = bt6.getText().toString();

b7 = bt7.getText().toString();

b8 = bt8.getText().toString();

b9 = bt9.getText().toString();

// These for Horizontal

if(b1.equals(b2) && b2.equals(b3) && !b1.equals(" ")){

Toast.makeText(this,"Winner is: "+b1,Toast.LENGTH\_SHORT).show();

newGame();

}else if (b4.equals(b5) && b5.equals(b6) && !b4.equals(" ")){

Toast.makeText(this,"Winner is: "+b4,Toast.LENGTH\_SHORT).show();

newGame();

}else if(b7.equals(b8) && b8.equals(b9) && !b7.equals(" ")){

Toast.makeText(this,"Winner is: "+b7,Toast.LENGTH\_SHORT).show();

newGame();

// These for Vertical

} else if (b2.equals(b5) && b5.equals(b8) && !b2.equals(" ")){

Toast.makeText(this,"Winner is: "+b2,Toast.LENGTH\_SHORT).show();

newGame();

}else if(b3.equals(b6) && !b6.equals(b9) && !b3.equals(" ")){

Toast.makeText(this,"Winner is: "+b3,Toast.LENGTH\_SHORT).show();

newGame();

}else if(b1.equals(b4) && b4.equals(b7) && !b1.equals(" ")){

Toast.makeText(this,"Winner is: "+b1,Toast.LENGTH\_SHORT).show();

newGame();

// These for diagonals

}else if(b1.equals(b5) && b5.equals(b9) && !b1.equals(" ")){

Toast.makeText(this,"Winner is: "+b1,Toast.LENGTH\_SHORT).show();

newGame();

}else if(b3.equals(b5) && b5.equals(b7) && !b3.equals(" ")){

Toast.makeText(this,"Winner is: "+b3,Toast.LENGTH\_SHORT).show();

newGame();

}}}

public void newGame(){

bt1.setText(" ");

bt2.setText(" ");

bt3.setText(" ");

bt4.setText(" ");

bt5.setText(" ");

bt6.setText(" ");

bt7.setText(" ");

bt8.setText(" ");

bt9.setText(" ");

count = 0;

flag = 0;

}}

* **Intents:** Intent is a class which is used to shifting of activities (i.e, changing of activities) in an app is called Intent. Intents are of 3 types:
* Explicit Intent
* Implicit Intent
* Broadcast Intent
* **What is an App?**

An app is a collection/group/set of activities is called an app.

An example for Intents:

# **About Firebase as Backend**

* **Firebase Realtime Database**: It is a new **NoSQL** database that allows you to store and synchronize data in real time, basically it will help you to build real time applications such as chat, collaboration and social media etc. without any need of complex servers. Data is stored as JSON and synchronized in real-time to client(user).
* **Firebase Authentication:** Firebase Authentication provides backend services SDKs, and UI libraries to authenticate users. It implements user signups, logins and more with ease and it supports authentication using passwords, phone numbers, Google, Facebook, Instagram, Twitter etc. It leverages industry standards like OAuth 2.0 and OpenID Connect.
* **Firebase Storage:** Cloud Storage for Firebase is a powerful simple and cost-effective object storage service. It adds Google security to a file uploads and downloads for your Firebase apps.

Client SDK’s stores images, audios, videos, document or user generated content. Admin SDK to manage buckets and create download URI’s.

* **Firebase Firestore:** Cloud Fire store is a scalable database for mobile, web and server development. It keeps your data in sync through Realtime listeners and offers offline support.
* Cloud Firestore uses data synchronization to update data on any connected device. Cloud Firestore is a cloud-hosted NoSQL database.

dp\_density pixels:width and height

sp-scalable pixel:font regarding we use

px-pixels

dip-density independent pixels

in-Inches

mm-millimeter

pt-points