ANDROID TUTORIAL

INTRODUCTION TO ANDROID PROGRAMMING

L - 1

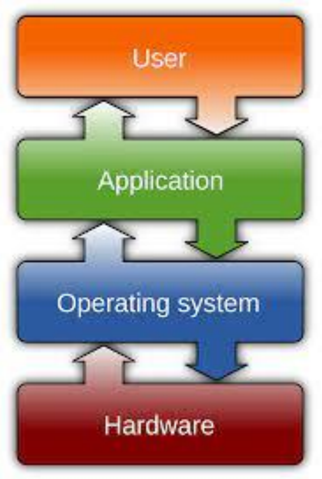
3rd Sem, MCA

CONTENT

- Introduction to Mobile app development
- Android Architecture,
- Android Studio Environment,
- Android Programming,
- UI components: Toast, Button.
- Activity Lifecycle.

INTRODUCTION

- **Android** is a mobile operating system; development led by Google (2007).
- Open source and Linux-based operating system for mobile devices.
- Its considered as a mobile operating system (But not limited to mobile only). Its currently used in various devices (mobiles, tablets, televisions etc).
  - **Open-Source**: any program whose source code is made freely available for use or **modification**.
  - Unlike proprietary software, open source software is computer software that is developed as a public, open collaboration and made freely available to the public.
  - **Operating system** (OS) is system software that manages computer hardware, software resources, and provides common services for computer programs.
  - **Linux** is a family of open-source Unix-like operating systems based on the Linux kernel.



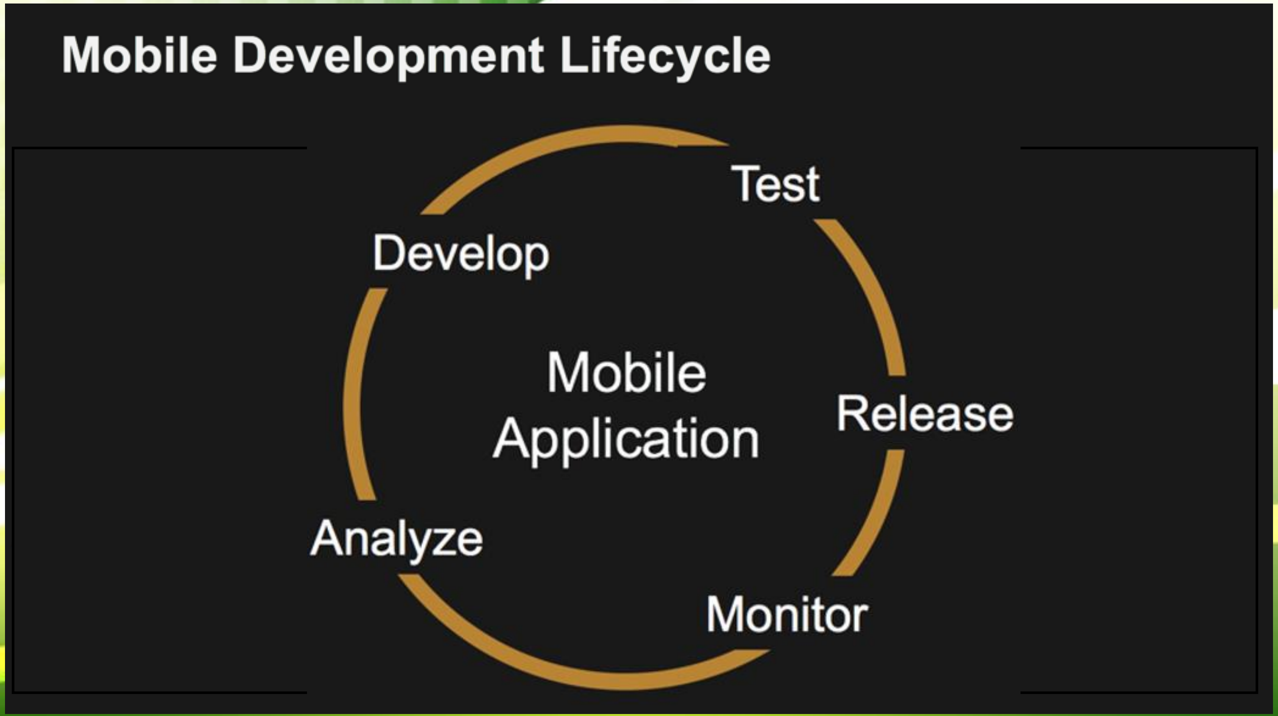
Parameter	Linux	Windows
Access	**Users can access the source code of kernel in Linux and can alter the kernel according to need.**	Usually, Users cannot access the source code.
Variety	**Linux has several distributions that are highly customizable.**	Windows have fewer options to customize.
Command-line	**(Terminal)** most useful tool of Linux system. Used for administration and daily tasks.	Windows command line is not such effective as compared to Linux terminal. Most users prefer GUI options for daily tasks.
Installation	Installation process is a bit complicated to set up as it requires many user inputs. Takes less time than Windows to install.	Windows OS is easy to install; requires fewer user input options during installation. Takes more time to install.
Ease of use	Meant for technical users with exposure to Linux commands. Troubleshooting process is also complicated as compared to Windows.	**Simple and rich GUI options (easy to use).** **Can be used by technical & non-technical users.** **Troubleshooting process is also much easy than Linux.**
Written in	Written in assembly language and C.	Written in C++ and assembly language.
Reliability	**Highly reliable and secure. Well-established system security, process management, and uptime.**	Not much reliable or secure as Linux. Easy target for malware & virus. Most vulnerable without anti-virus.
Support	Linux has a good support as it has a huge community of user forums and online search.	Windows also provide good support to user. Provides paid & free support through online forum.
Update	Provides full control to its users on updates. User can install update whenever needed. Takes less time to install an update.	Updates are not much in user control. Take too much time to install.
Fee	**Open source operating system (free of cost).**	Not the open source operating system (license fee)
Efficiency	**More efficient in comparison of windows.**	Less efficient.
Kernel	Monolithic kernel is used.	Micro & hybrid kernel is used.

KERNEL

- Central component of OS that manages operations (Facilitates interactions) between hardware & software.
- Helps with process, memory, disk, task, CPU time management; file systems, device control, and networking.
- Acts as bridge between applications and data processing performed at hardware level using inter-process communication and system calls.
  - **Monolithic Kernel :** all OS services operate in kernel space. *Example : Unix, Linux, etc.*
  - **Micro Kernel –** minimalist approach. It has virtual memory and thread scheduling. It is more stable with less services in kernel space. It puts rest in user space. *Example : Mach, L4, AmigaOS, Minix, K42 etc.*
  - **Hybrid Kernel –** combination of monolithic and micro kernel. It has speed and design of monolithic kernel and modularity and stability of microkernel. *Example : Windows NT, Netware, BeOS etc.*
  - **Exo Kernel –** follows end-to-end principle, with fewest hardware abstractions.
  - **Nano Kernel –** offers hardware abstraction but without system services.

MOBILE APPLICATION DEVELOPMENT

- Set of processes and procedures involved in writing software for small, wireless computing devices.
  - Like Web application development, mobile application development has its roots in more traditional software development.
  - Mobile apps are often written specifically to take advantage of the unique features a particular mobile device offers.



MOBILE APPLICATION DEVELOPMENT

Three basic types of mobile apps;

Native apps

- Built specifically for one specific operating system, i.e. native Android mobile apps or native iOS apps.
- Because they're built for just one platform, you cannot mix and match
- i.e. use a Blackberry app on an Android phone or use an iOS app on a Windows phone.

Web apps (Progressive Web Apps / PWAs)

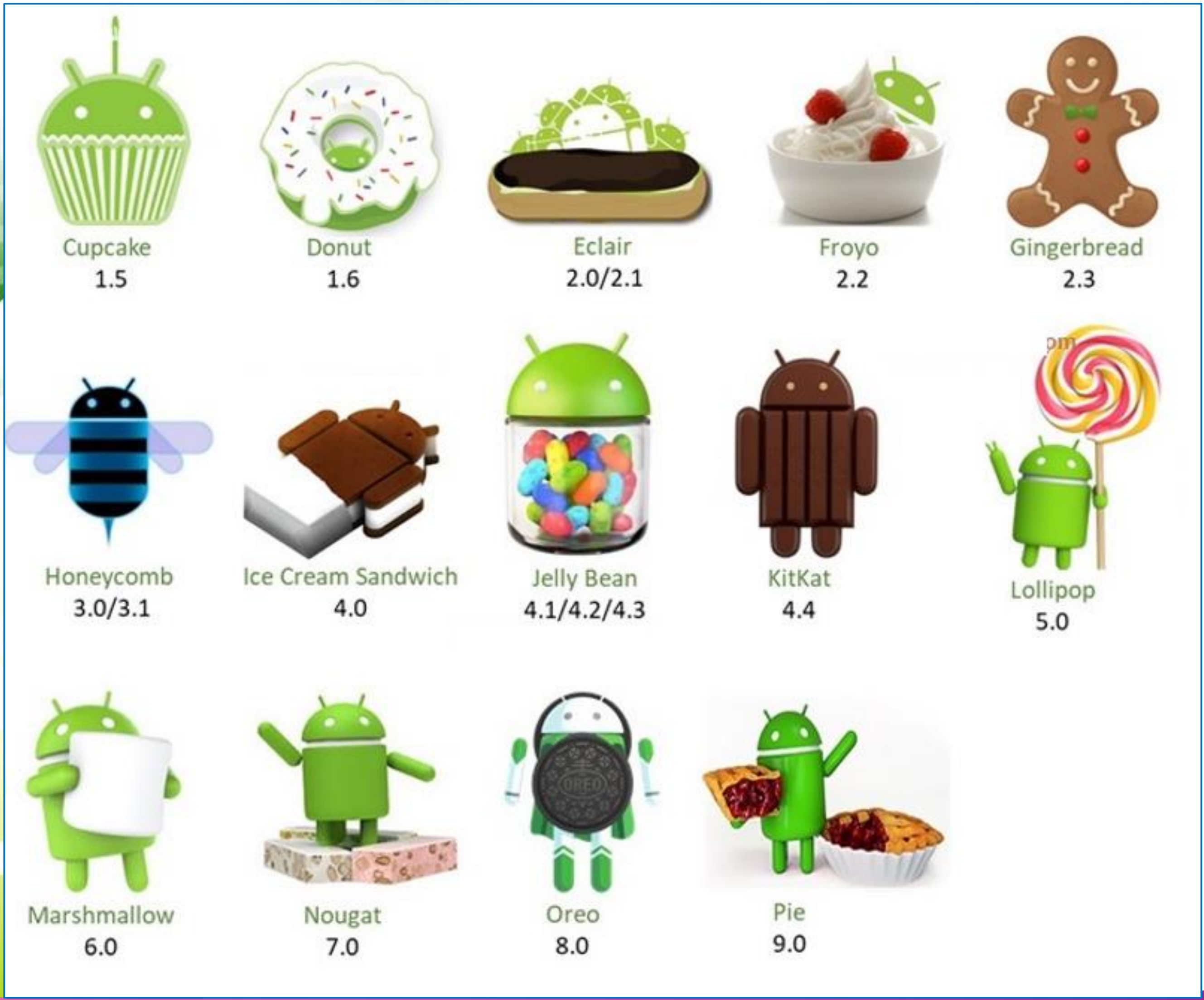
- Behave similarly to native apps, but are accessed via a web browser on mobile device.
- Responsive versions of websites that can work on any mobile device or OS.
- They're not standalone apps, i.e. No need to download and install code into your device.
- They're actually responsive websites that adapt its user interface to the device. In fact, when comes to "install" a web app, it often simply bookmarks the website URL on the device.

Hybrid apps

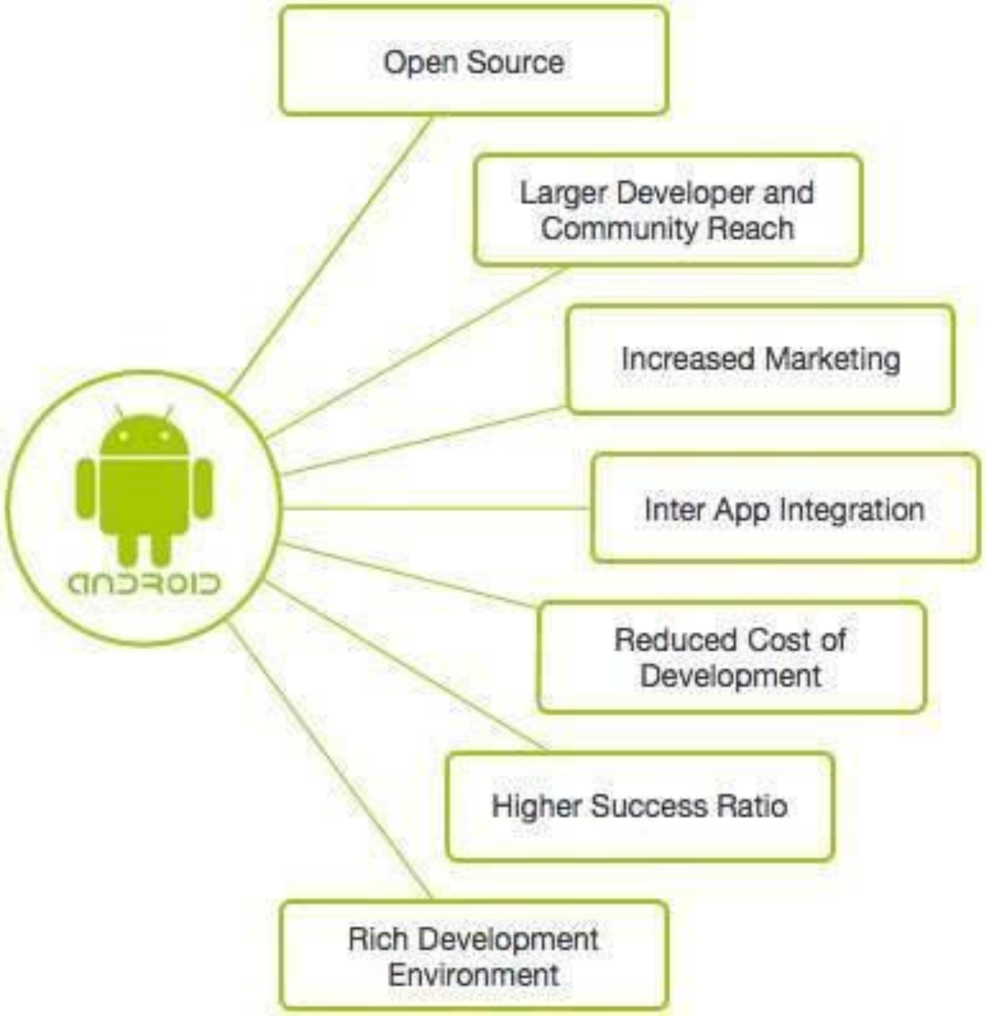
- Combinations of both native and web apps, but wrapped within a native app.
- Gives it the ability to have its own icon or be downloaded from an app store.
- They might have a home screen app icon, responsive design, fast performance, even be able to function offline, but they're really web apps made to look native.

ANDROID EVOLUTION

- Code names of android ranges from A to Z.



ANDROID FEATURES


Open Source
Larger Developer and Community Reach
Increased Marketing
Inter App Integration
Reduced Cost of Development
Higher Success Ratio
Rich Development Environment
android

- Open-source.
- Anyone can customize the Android Platform.
- A lot of mobile applications (huge demand by user).
- Provides huge range of interesting, customizable features to make use of.

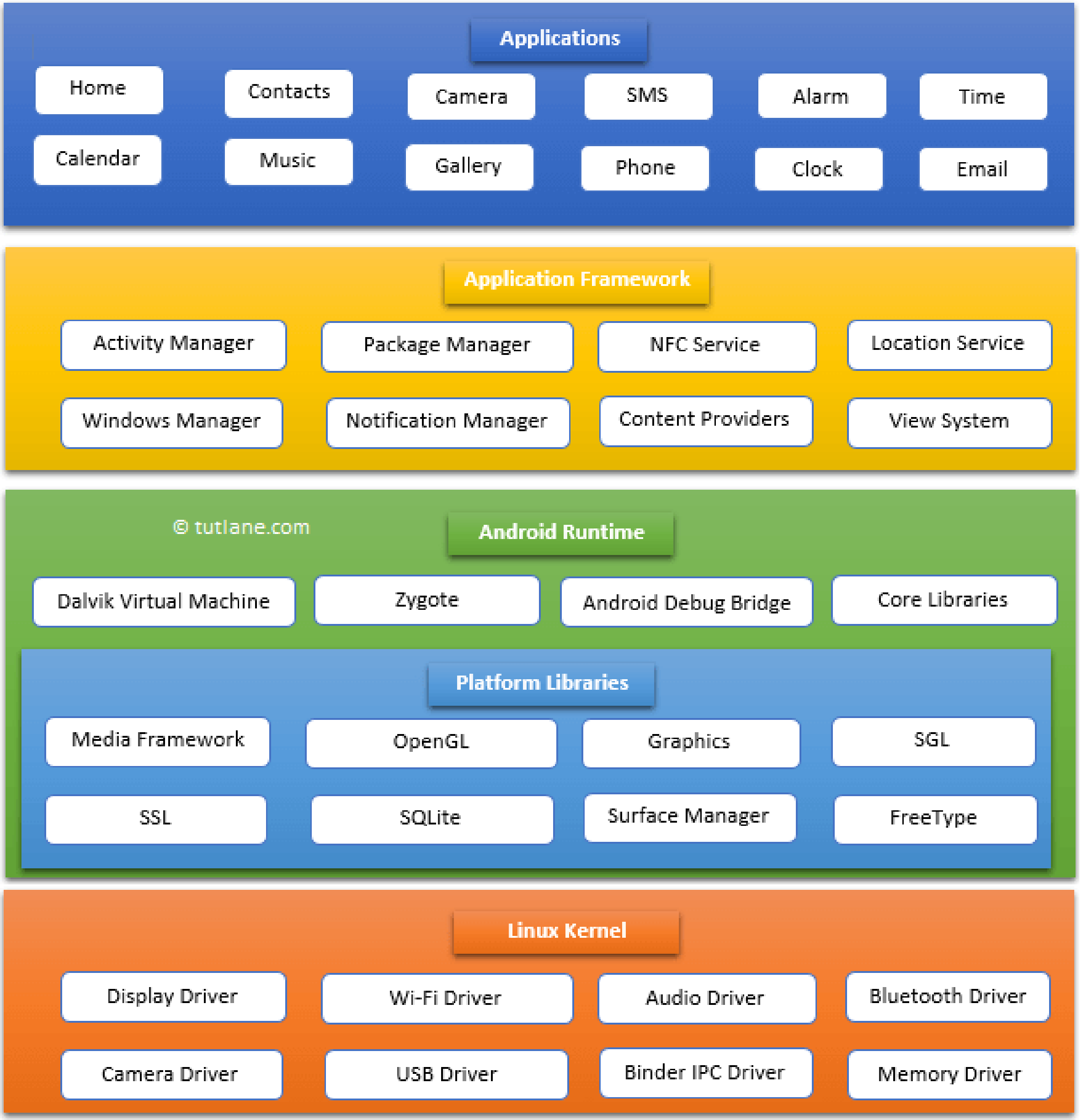
ANDROID FEATURES

- **User Interface** – Android operating system provides a UI which is associated with activity and presented to user.
- **Messaging** – It supports messaging services -SMS , MMS.
- **Web Browser** – It is based on the open-source WebKit layout engine which is coupled with Chrome's V8 JavaScript engine that supports HTML5 and CSS3.
- **Connectivity** – It has various connections such as Bluetooth, Wi-Fi, GSM/EDGE, CDMA etc.
- **Storage** – A light relational database – SQL (SQlite) is used for the storage purpose.
- **Multi-touch** – Android has native support for multi-touch feature.
- **Multi-tasking** – Android support multi tasking where user can perform multiple tasks simultaneously.
- **Multi-Language** – It supports single direction and bi-directional text feature.
- **Wi-Fi Direct** – Android supports a technology that lets apps discover and pair directly over a high-bandwidth peer-to-peer connection.
- **Android Beam** – This feature is used to share instantly just by touching two NFC enabled phones together.

ARCHITECTURE

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

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



LINUX KERNEL

- Linux kernel is the bottom most layer in the architecture of android.
- It never really interacts with the users and developers, but is at the heart of the whole system.
- Provides level of abstraction from device hardware.
- Contains all essential hardware drivers like camera, keypad, display etc.
- Responsible for device management, resource access, memory management, security settings, power management, network stack etc.

PLATFORM LIBRARIES

- On the top of linux kernel, their are Native libraries.
- Libraries carry a set of instructions to guide the device to handle different types of data.
- WebKit, OpenGL, FreeType, SQLite, Media, C runtime library (libc) etc.
  - WebKit library is responsible for browser support,
  - SQLite is for database,
  - FreeType for font support,
  - Media Framework for playing and recording audio and video formats,
  - SSL libraries responsible for Internet security,
  - Surface manager responsible for managing access to the display subsystem.
  - SGL and OpenGL both cross-language, cross-platform application program interface (API) are used for 2D and 3D computer graphics etc.

PLATFORM LIBRARIES

- **android.app** − Provides access to the application model and is the cornerstone of all Android applications.
- **android.content** − Facilitates content access, publishing & messaging between applications & application components.
- **android.database** − Used to access data published by content provider & include SQLite database management classes.
- **android.opengl** − A Java interface to the OpenGL ES 3D graphics rendering API.
- **android.os** − Provides applications with access to standard operating system services including messages, system services and inter-process communication.
- **android.text** − Used to render and manipulate text on a device display.
- **android.view** − The fundamental building blocks of application user interfaces.
- **android.widget** − A rich collection of pre-built user interface components such as buttons, labels, list views, layout managers, radio buttons etc.
- **android.webkit** − A set of classes intended to allow web-browsing capabilities to be built into applications.

ANDROID RUNTIME

- Runtime provides a key components like core libraries and Dalvik Virtual Machine (DVM).
- DVM is responsible to run android application.
- Provides base for the application framework and powers the application with help of core libraries.
- DVM consumes less memory and provides fast performance.
- Core libraries enable to implement android applications using standard JAVA or Kotlin programming languages.
- DVM is like JVM (Java Virtual Machine) but specially designed and optimized for mobile devices.

JAVA VIRTUAL MACHINE (JVM)

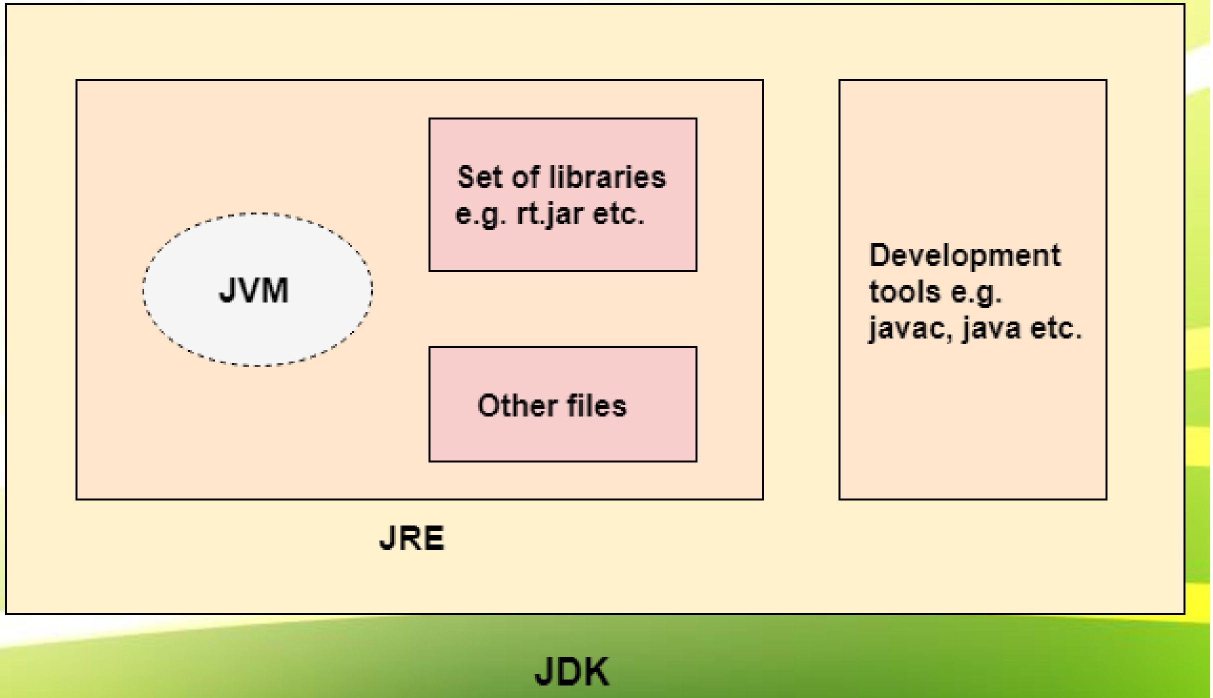
- JVM is a virtual machine that provides runtime environment in which java bytecode can be executed.
- Its implementation is known as JRE (Java Runtime Environment).
- Whenever you write java command on command prompt to run java class, an instance of JVM is created.
- JVM performs following operation:
  - Loads code
  - Verifies code
  - Executes code
  - Provides runtime environment
- JVM provides definitions for the Memory area, Class file format, Register set, Garbage-collected heap, Fatal error reporting etc.

JAVA RUNTIME ENVIRONMENT (JRE)

- JRE is used to provide the runtime environment for Java applications.
- It is the implementation of JVM. It contains a set of libraries + other files that JVM uses at runtime.

Java Development Kit (JDK)

- JDK is a software development environment which contains JRE + development tools.
- JDK contains a private JVM and a few other resources (interpreter/loader, compiler, archiver, documentation generator, etc.) to complete Java Application development.


JVM
Set of libraries e.g. rt.jar etc.
Other files
Development tools e.g. javac, java etc.
JRE
JDK

APPLICATION FRAMEWORK

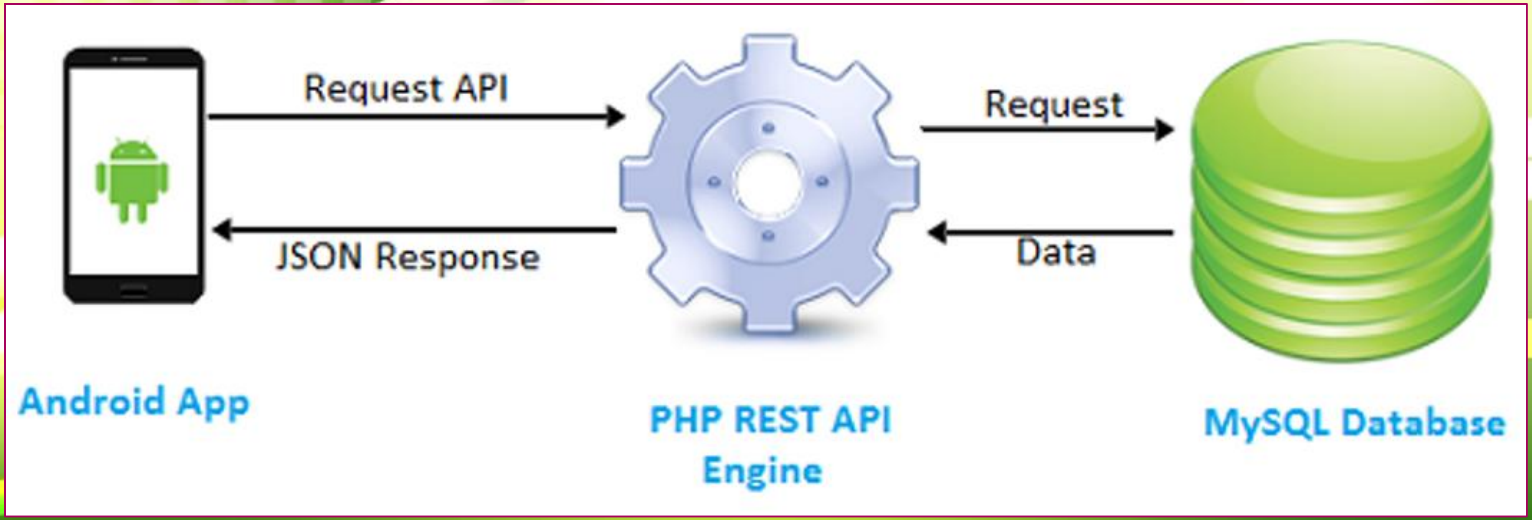
- The applications directly interact with these blocks of Android architecture.
- These programs manage basic functions of phone (resource management, voice call management etc.)
- It provides many higher-level services to applications in form of Java classes.
- Android framework includes **Android API's;** UI, telephony, resources, locations, package managers, activity management, content providers, resource manager, notification manager, view system.
  - **Activity Manager** − Controls all aspects of the application lifecycle and activity stack.
  - **Content Providers** − Allows applications to publish and share data with other applications.
  - **Resource Manager** − Provides access to non-code embedded resources such as strings, color settings and user interface layouts.
  - **Notifications Manager** − Allows applications to display alerts and notifications to the user.
  - **View System** − An extensible set of views used to create application user interfaces.

APPLICATIONS

- Applications are at the topmost layer of the Android stack.
- All applications will be installed in this layer only.
    - Pre-installed applications like home, contacts, camera, gallery etc and
    - third party applications downloaded from the play store like chat applications, games etc.
- Runs within the Android run time with help of classes and services provided by application framework.

APPLICATION PROGRAM INTERFACE (API)

- API is used in **mobile apps.**
- API is particular set of rules (code) & specifications that programs follow to communicate with each other.
  - For sending and receiving data from/to the server, a middle-man is needed who is platform independent.
  - That middleman handles the requests and serves the response to user.
  - End user sends a request, API executes the instruction to get the data from server and responds back to the user.



DEVELOPMENT ENVIRONMENT

- Android application development supported on most operating systems – Windows, Mac, Linux, etc.
- Android supports java, c++, c# etc. language to develop android applications.
  - Java is the officially supported language for android.
- Most of the required tools to develop Android applications are freely available.
- Basic Environment Requirement:
  - Java JDK5 or later version
  - Android Studio (*or Eclipse IDE & Android SDK*)

SOFTWARE DEVELOPMENT KIT (SDK)

- Android SDK is a collection of libraries and Software Development tools used to develop applications for Android platform.
- Whenever Google releases a new version/update of Android, a corresponding SDK also releases with it.
- Android SDK includes the following:
  - Required libraries.
  - Debugger.
  - Emulator.
  - Relevant documentation for Android application program interfaces (APIs).
  - Sample source code.
  - Tutorials for the Android OS.

ENVIRONMENT SETUP

Install Java Development Kit (JDK)

- https://www.oracle.com/in/java/technologies/javase-jdk15-downloads.html
- System Environment variable → Java path setting

Install Android Studio

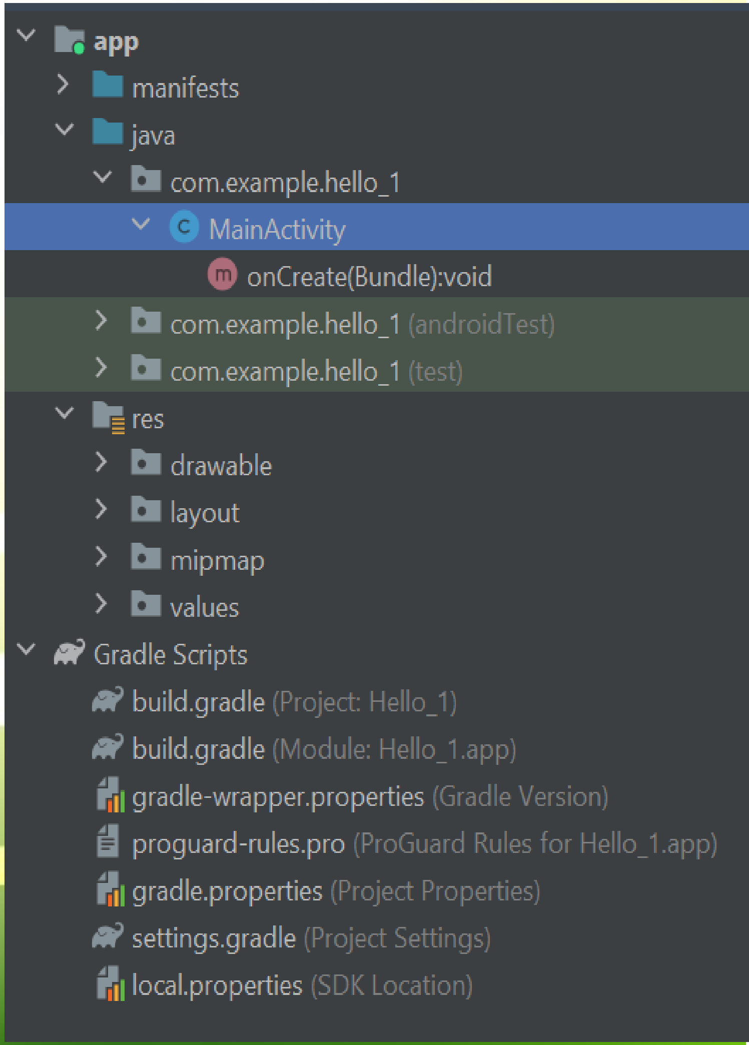
- https://developer.android.com/studio
- Setup emulator & libraries

Phone Developer debugging setting

- phone → setting → about phone → software information → build number (4-7 taps) → phone lock permission
- setting → system → developer option → enable USB debugging

ANATOMY OF ANDROID APPLICATION

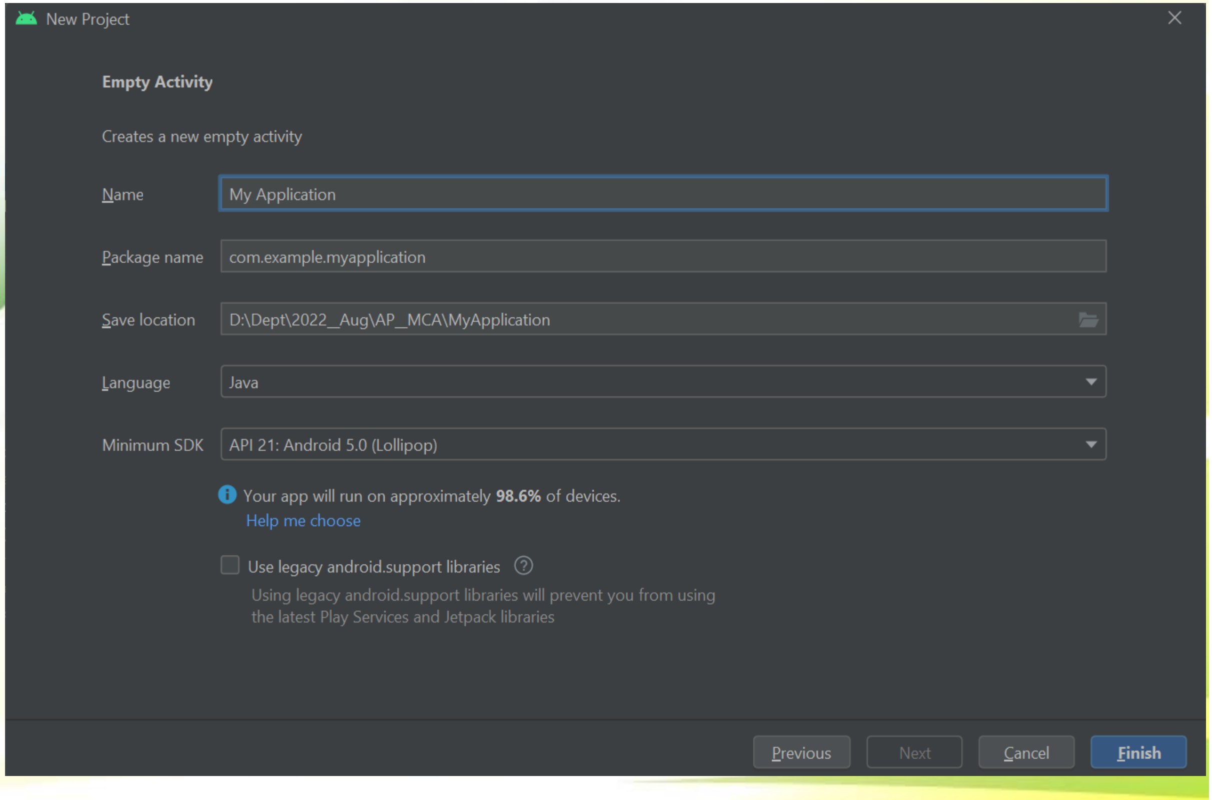
- **App** describes fundamental characteristics of app and defines each of its components.
- **Java** contains the .java source files for the project.
  - By default, it includes an MainActivity.java source file.
  - Under this, create all activities which have .java extensions and all code behind the application.
  - MainActivity.java is actual file which gets converted to dalvik executable and runs app.
- **Res** is a directory for files that define app's UI.
  - Add TextView, button etc to build GUI and use its various attributes like android:layout_width, android:layout_height etc which are used to set its various features (width and height).
- **Gradle scripts** is an auto generated file which contains compileSdkVersion, buildToolsVersion, applicationId, minSdkVersion, targetSdkVersion, versionCode and versionName.



```
app
  manifests
  java
    com.example.hello_1
      MainActivity
        onCreate(Bundle):void
    com.example.hello_1 (androidTest)
    com.example.hello_1 (test)
  res
    drawable
    layout
    mipmap
    values
Gradle Scripts
  build.gradle (Project: Hello_1)
  build.gradle (Module: Hello_1.app)
  gradle-wrapper.properties (Gradle Version)
  proguard-rules.pro (ProGuard Rules for Hello_1.app)
  gradle.properties (Project Properties)
  settings.gradle (Project Settings)
  local.properties (SDK Location)
```

FIRST PROGRAM

- In Android Studio, choose **File → New Project.**
- Enter **Hello_Android** as the application name.
- Check Package Name.
- Choose a location for your project.
- Choose Java language.
- Select Phone and Tablet, choose a Minimum SDK version
- In the Create Activity box, choose Activity and click Next.
- Click the Finish button.



New Project

Empty Activity

Creates a new empty activity

Name: My Application

Package name: com.example.myapplication

Save location: D:\Dept\2022_Aug\AP_MCA\MyApplication

Language: Java

Minimum SDK: API 21: Android 5.0 (Lollipop)

Your app will run on approximately **98.6%** of devices.

Help me choose

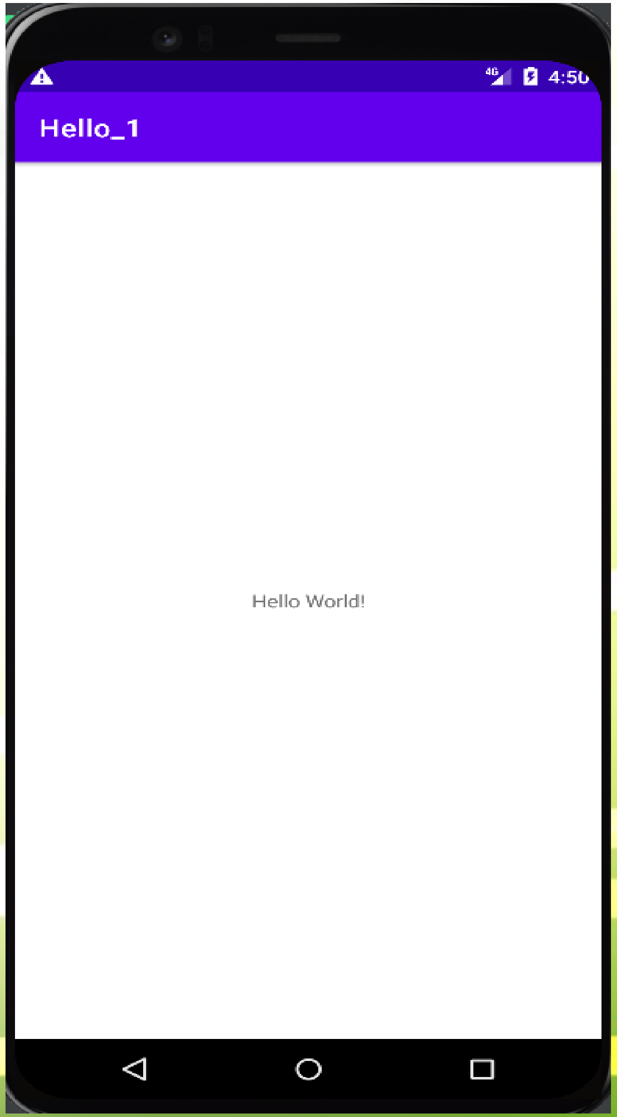
□ Use legacy android.support libraries

Using legacy android.support libraries will prevent you from using the latest Play Services and Jetpack libraries

OR

EXAMPLE

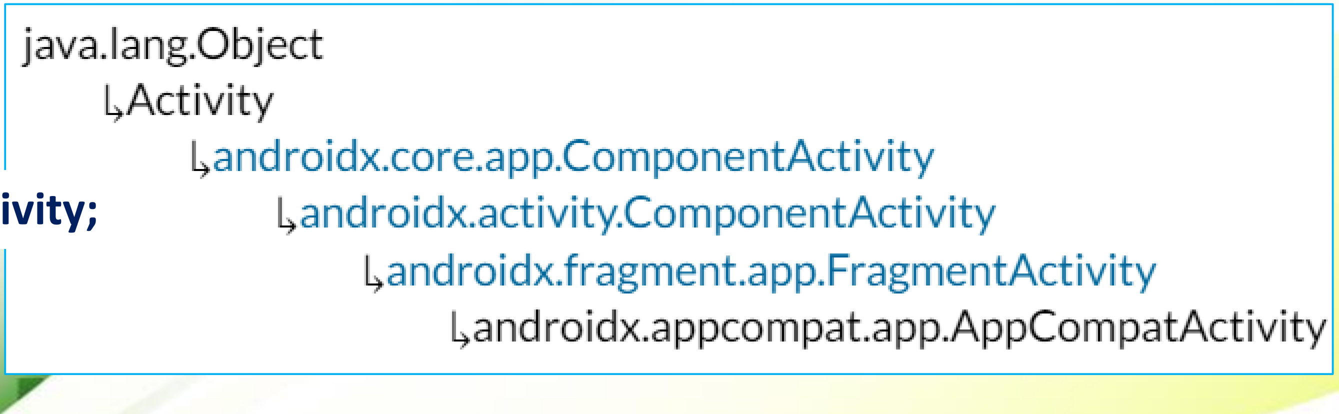
```
package com.example.hello_1;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;

public class MainActivity extends AppCompatActivity
{
   @Override
   protected void onCreate(Bundle savedInstanceState)
{
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
   }
}
```



Note: @Override is a Java annotation.
It tells the compiler that the following method overrides a method of its superclass.

FIRST PROGRAM

```
import androidx.appcompat.app.AppCompatActivity;
```

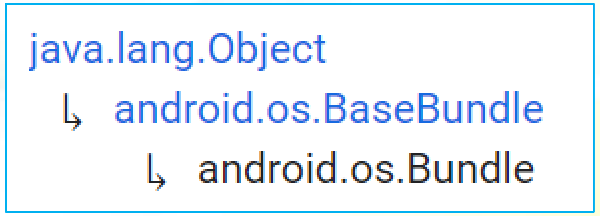


```
java.lang.Object
   ↳Activity
      ↳androidx.core.app.ComponentActivity
         ↳androidx.activity.ComponentActivity
            ↳androidx.fragment.app.FragmentActivity
               ↳androidx.appcompat.app.AppCompatActivity
```

- Base class for activities.
- androidx packages fully replace the Support Library by providing newer platform features and new libraries.
- Other libraries have migrated to use the AndroidX namespace libraries, including Google Play services, Firebase, Butterknife, Databinding Mockito, and SQLDelight among others

FIRST PROGRAM

```
import android.os.Bundle;
```



java.lang.Object
   ↳ android.os.BaseBundle
      ↳ android.os.Bundle

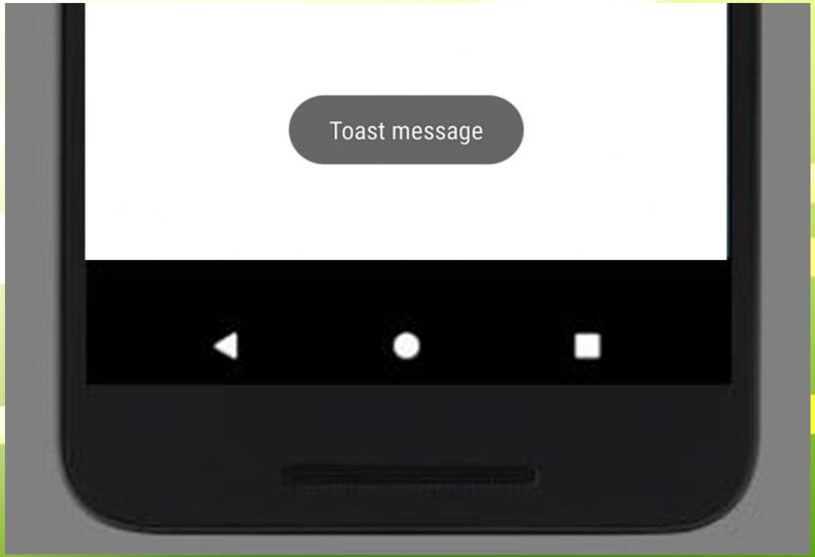
- Bundle is a utility class that lets to store a set of name-value pairs.
- Android Bundle is used to pass data between activities.
- Bundles are used with intent and values are sent and retrieved in the same fashion, as it is done in the case of intent.
- It depends on the user what type of values the user wants to pass, but bundles can hold all types of values (int, String, boolean, char) and pass them to the new activity.
- Both onCreate() and onFreeze() methods take Bundle as a parameter.
- A bundle object is ideal for storing a set of state variables in the onFreeze method; and the same state variables can be read back in the onCreate method.

LAB - 1

L1	1. Create a **Hello world** android application using android studio and compile it using android SDK. Run it on both different Android Virtual Devices (AVD) and your own mobile. 2. Create an android application to display the welcome message as a **toast**. (A) Default toast display (B) Toast on button click 3. Create an android application and understand the application **life-cycle** in different scenarios.

TOAST

- Andorid Toast can be used to display information for a short period of time.
- A toast contains message to be displayed quickly and disappears after sometime.
- *android.widget.Toast* class is the subclass of *java.lang.Object* class.
    - Toast class is used to show notification for a particular interval of time.
    - After sometime it disappears.
    - It doesn't block the user interaction.



CONSTANTS OF TOAST CLASS

There are only 2 constants of Toast class.

Constant	Description
public static final int **LENGTH_LONG**	displays view for the long duration of time.
public static final int **LENGTH_SHORT**	displays view for the short duration of time.

METHODS OF TOAST CLASS

Widely used methods of Toast class are.

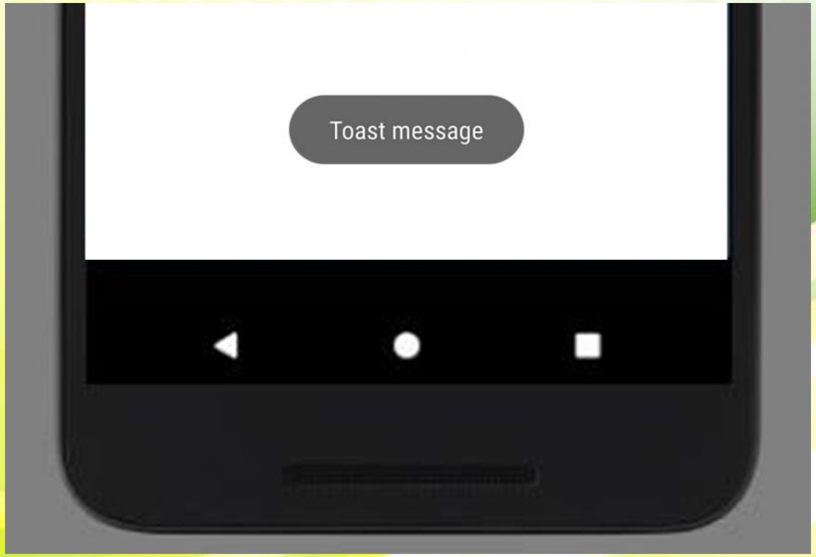
Method	Description
public static Toast **makeText**(Context context, CharSequence text, int duration)	makes the toast containing text and duration.
public void show()	displays toast.
public void **setMargin** (float horizontalMargin, float verticalMargin)	changes the horizontal and vertical margin difference.

Example:

```
Toast toast=Toast.makeText(getApplicationContext(),"Hello Javatpoint",Toast.LENGTH_SHORT);
toast.setMargin(50,50);
toast.show();
```

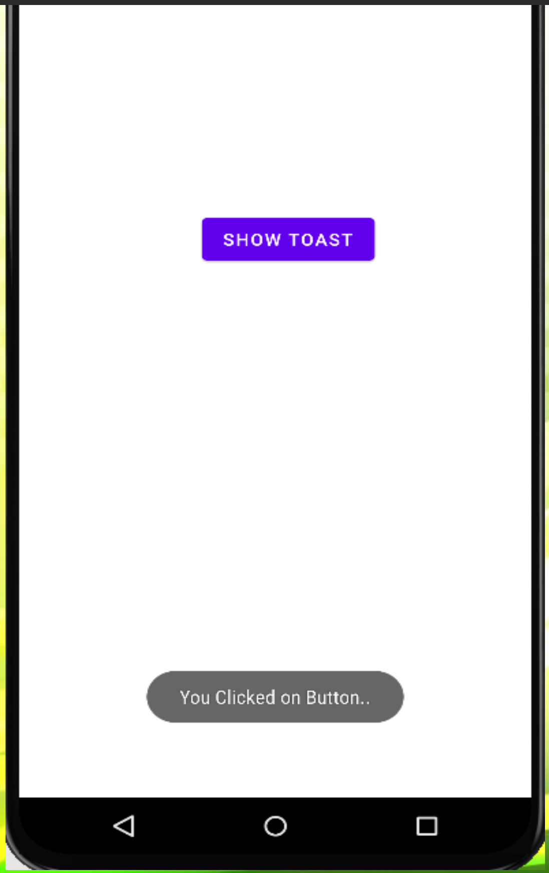
EXAMPLE

File: MainActivity.java



```
package example.android.toast;

import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        //Displaying Toast with "Toast Message" message
       Toast.makeText(getApplicationContext(),"Toast Message",
                                  Toast.LENGTH_SHORT).show();
    }
}
```

```
<Button
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerInParent="true"
    android:text="Show Toast"
    android:onClick="toast"/>
```



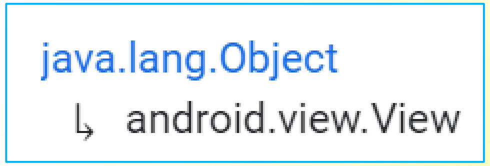
You Clicked on Button..

```
package com.example.toast;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    public void toast(View view)
    {
        Toast.makeText(MainActivity.this, "You Clicked on
                Button..",Toast.LENGTH_SHORT).show();
    }
}
```

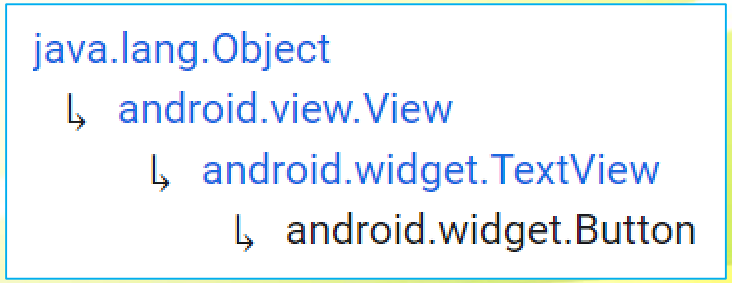
FIRST PROGRAM

```
import android.view.View;
import android.widget.Button;
import android.widget.Toast;
```

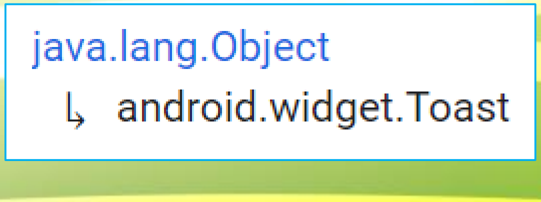
- View class represents the basic building block for user interface components.
- A View occupies a rectangular area on the screen and is responsible for drawing and event handling.
- View is the base class for widgets, which are used to create interactive UI components (buttons, text fields, etc.).
- ViewGroup subclass is base class for layouts, which are invisible containers that hold other Views (or other ViewGroups) and define their layout properties.



```
java.lang.Object
   ↳ android.view.View
```



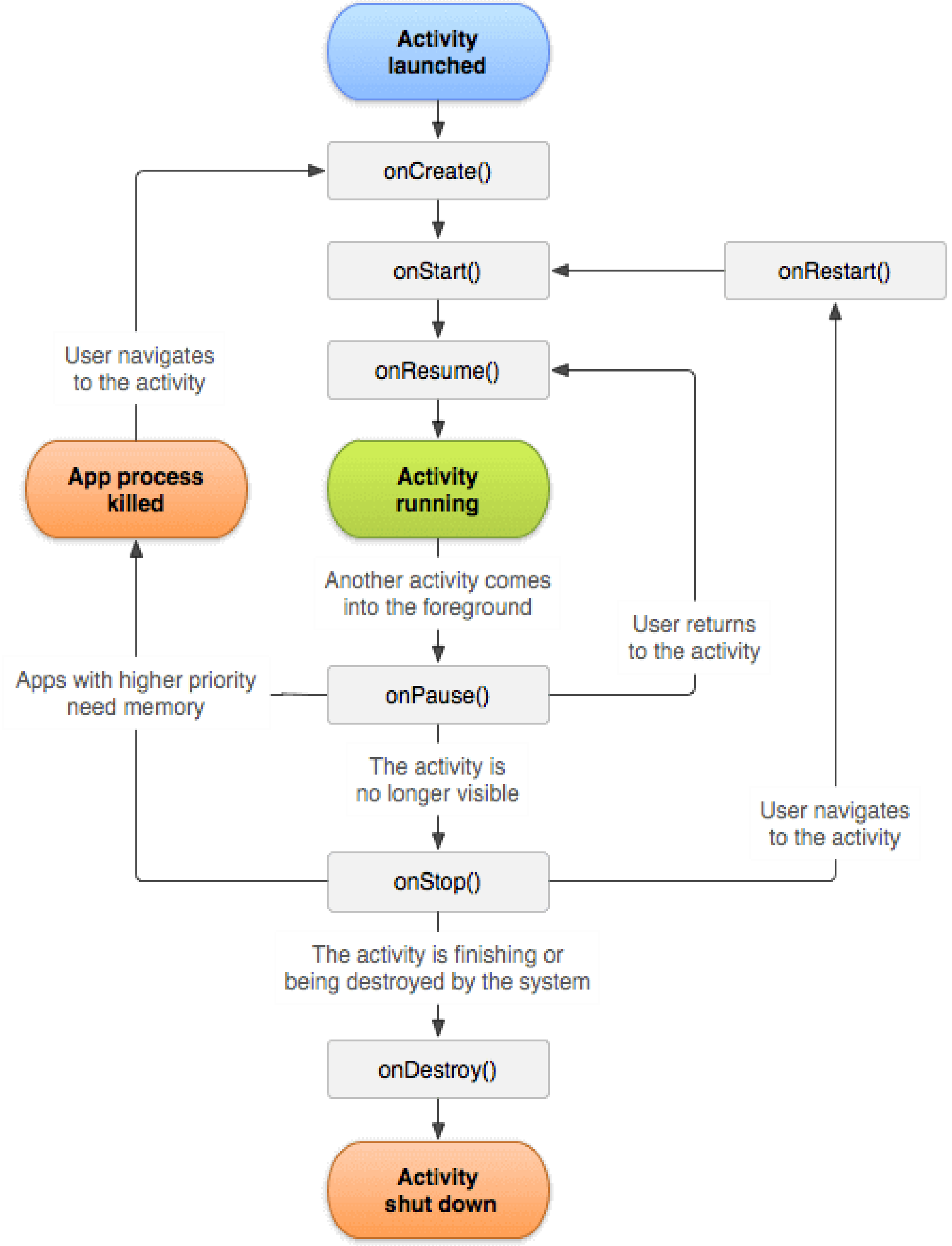
```
java.lang.Object
   ↳ android.view.View
        ↳ android.widget.TextView
             ↳ android.widget.Button
```



```
java.lang.Object
   ↳ android.widget.Toast
```

ACTIVITY LIFECYCLE

Method	Description
onCreate()	called when activity is first created.
onStart()	called when activity is becoming visible to user.
onResume()	called when activity will start interacting with user.
onPause()	called when activity is not visible to the user.
onStop()	called when activity is no longer visible to user.
onRestart()	called after your activity is stopped, prior to start.
onDestroy()	called before the activity is destroyed.


Activity launched
onCreate()
onStart()
onRestart()
onResume()
User navigates to the activity
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

EXAMPLE

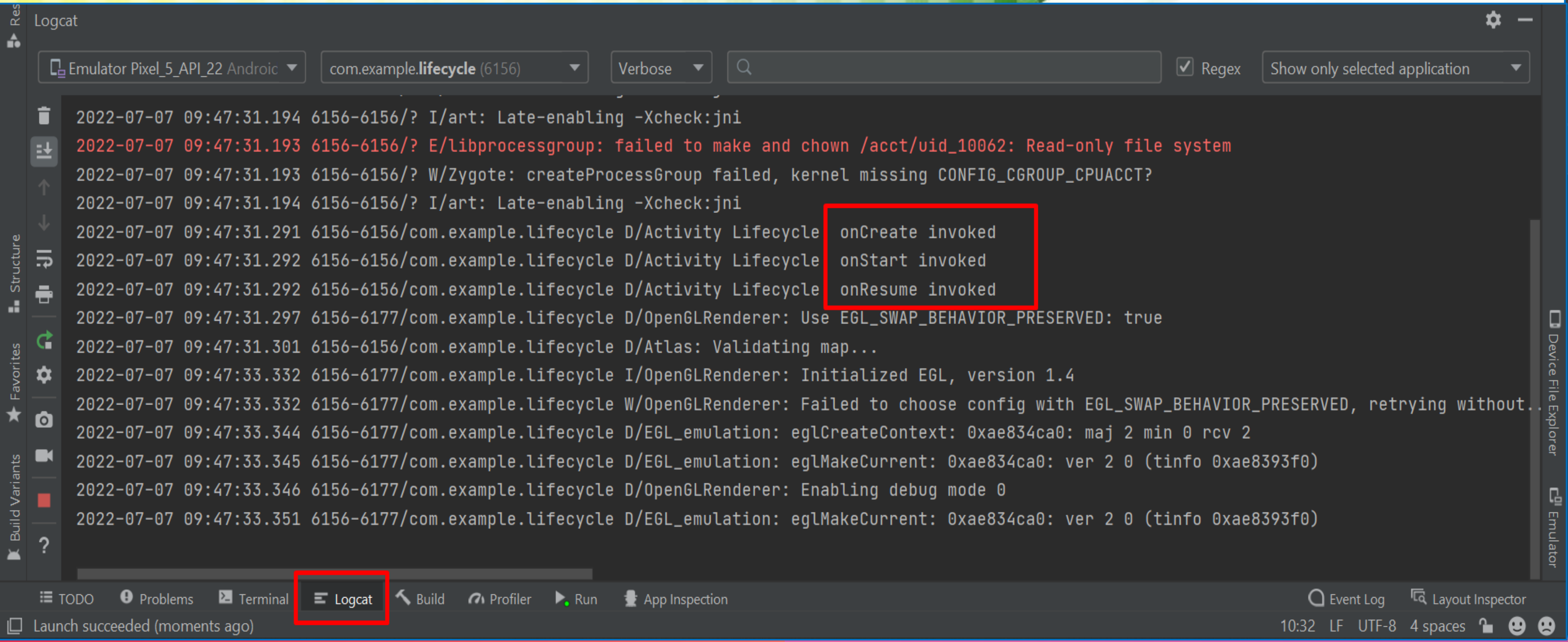
```
import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
```

```
  @Override
    protected void onCreate(Bundle savedInstanceState)
{
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Log.d("Activity Lifecycle","onCreate invoked");
    }
    @Override
    protected void onStart() {
        super.onStart();
        Log.d("Activity Lifecycle","onStart invoked");
    }
    @Override
    protected void onResume() {
        super.onResume();
        Log.d("Activity Lifecycle","onResume invoked");
    }
```

```
@Override
    protected void onPause() {
        super.onPause();
        Log.d("Activity Lifecycle","onPause invoked");
    }
    @Override
    protected void onStop() {
        super.onStop();
        Log.d("Activity Lifecycle","onStop invoked");
    }
    @Override
    protected void onRestart() {
        super.onRestart();
        Log.d("Activity Lifecycle","onRestart invoked");
    }
    @Override
    protected void onDestroy() {
        super.onDestroy();
        Log.d("Activity Lifecycle","onDestroy invoked");
    }
```

ACTIVITY LIFECYCLE

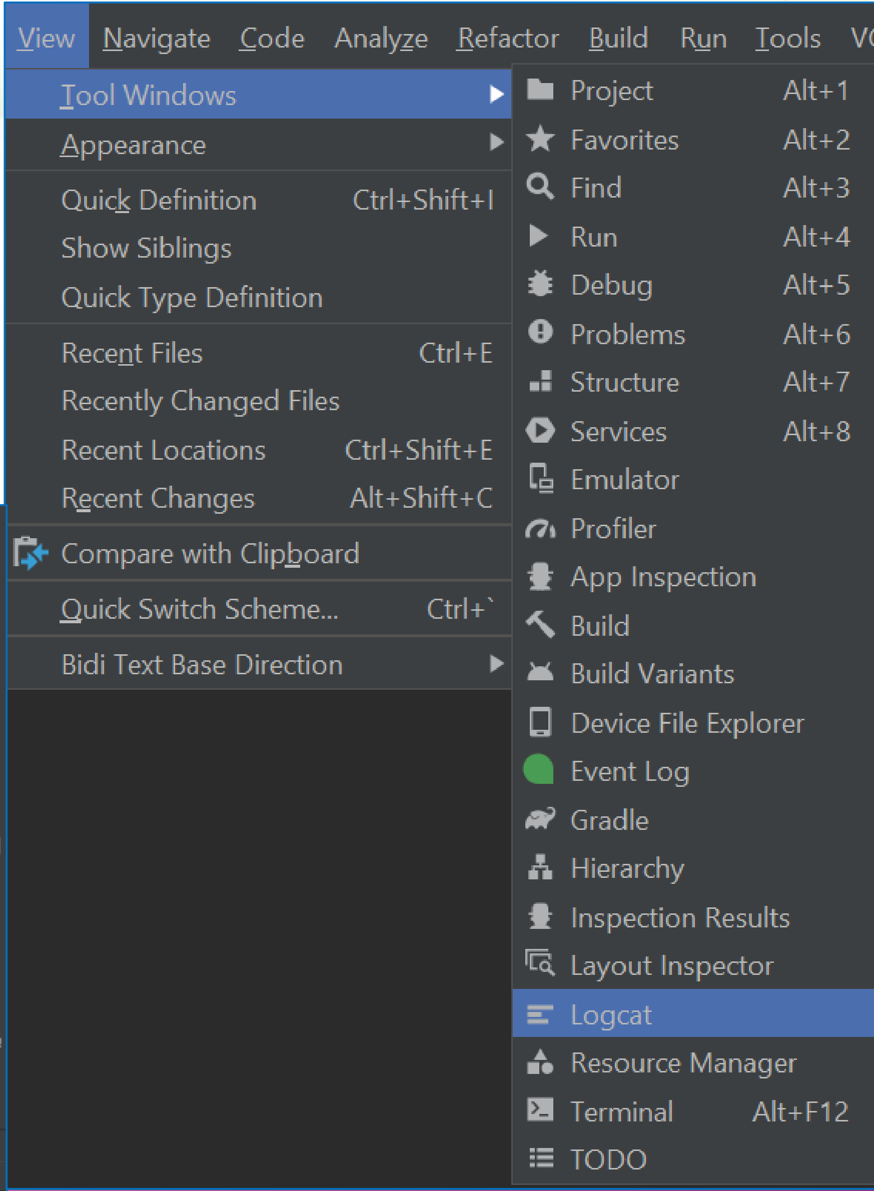
- Logcat is a command-line tool that dumps a log of system messages.
- Includes stack traces when device throws error; and messages that has been written from app with Log class.



Emulator Pixel_5_API_22 Androic | com.example.lifecycle (6156) | Verbose | ☑ Regex | Show only selected application

```
2022-07-07 09:47:31.194 6156-6156/? I/art: Late-enabling -Xcheck:jni
2022-07-07 09:47:31.193 6156-6156/? E/libprocessgroup: failed to make and chown /acct/uid_10062: Read-only file system
2022-07-07 09:47:31.193 6156-6156/? W/Zygote: createProcessGroup failed, kernel missing CONFIG_CGROUP_CPUACCT?
2022-07-07 09:47:31.194 6156-6156/? I/art: Late-enabling -Xcheck:jni
2022-07-07 09:47:31.291 6156-6156/com.example.lifecycle D/Activity Lifecycle: onCreate invoked
2022-07-07 09:47:31.292 6156-6156/com.example.lifecycle D/Activity Lifecycle: onStart invoked
2022-07-07 09:47:31.292 6156-6156/com.example.lifecycle D/Activity Lifecycle: onResume invoked
2022-07-07 09:47:31.297 6156-6177/com.example.lifecycle D/OpenGLRenderer: Use EGL_SWAP_BEHAVIOR_PRESERVED: true
2022-07-07 09:47:31.301 6156-6156/com.example.lifecycle D/Atlas: Validating map...
2022-07-07 09:47:33.332 6156-6177/com.example.lifecycle I/OpenGLRenderer: Initialized EGL, version 1.4
2022-07-07 09:47:33.332 6156-6177/com.example.lifecycle W/OpenGLRenderer: Failed to choose config with EGL_SWAP_BEHAVIOR_PRESERVED, retrying without...
2022-07-07 09:47:33.344 6156-6177/com.example.lifecycle D/EGL_emulation: eglCreateContext: 0xae834ca0: maj 2 min 0 rcv 2
2022-07-07 09:47:33.345 6156-6177/com.example.lifecycle D/EGL_emulation: eglMakeCurrent: 0xae834ca0: ver 2 0 (tinfo 0xae8393f0)
2022-07-07 09:47:33.346 6156-6177/com.example.lifecycle D/OpenGLRenderer: Enabling debug mode 0
2022-07-07 09:47:33.351 6156-6177/com.example.lifecycle D/EGL_emulation: eglMakeCurrent: 0xae834ca0: ver 2 0 (tinfo 0xae8393f0)
```

Launch succeeded (moments ago)



LAB-1

L1	1. Create a **Hello world** android application using android studio and compile it using android SDK. Run it on both different Android Virtual Devices (AVD) and your own mobile. 2. Create an android application to display the welcome message as a **toast**. (A) Default toast display (B) Toast on button click 3. Create an android application and understand the application **life-cycle** in different scenarios.