MOBILE APPLICATION DEVELOPMENT

prepared by

M. MOHAMED ZAMAM NAZAR

INTRODUCTION

You could argue that the success of the Microsoft Windows OS on the desktop, somewhere over 90 percent of all desktops, is attributable to Microsoft being at the right place at the right time. The rapid evolution and popularity of the personal computer, the rapid advance of CPU technology, and the standardization of PC architecture based on the IBM PC-AT gave Microsoft a market on which to thrive. This is not to say that Microsoft doesn't deserve credit for its success in its own right or that it didn't contribute to the success of the personal computer industry.

The same could be said for the rapid advance in technology and popularity of the smart phone and the evolution of the Android OS. In three short years, Android has become the most popular OS on mobile phones in the United States. In the spring of 2011, a Nielsen survey concluded that Android controlled 36 percent of the market, followed by Apple iOS at 26 percent (Feb-Apr 2011 Nielsen Mobile Insights, National). Estimates are that by the end of 2012, Android will control 50 percent of the global smart phone market, the other 50 percent being divided among all other systems. In February 2010, CBS reported that the worldwide number of mobile phone subscriptions was 4.6 billion, and that was expected to grow to5 billion during 2010. 2011 estimates are 5.6 billion mobile subscriptions worldwide, 77 percent of the population, with the largest growth in China and India. So, what is Android, and where did it come from?

In August 2005, Business Week reported that Google acquired Android, Inc., a 22-month-old start-up, which signaled Google's push into the wireless market. In 2007, Google and several other industry giants such as Motorola, Toshiba, Texas Instruments, and T-Mobile, just to name a few, formed the Open Handset Alliance. The alliance members released a significant amount of intellectual property into open source and released the Android platform.

In September 2008, T-Mobile released the G1, the first smart phone based on Android. It ran Android 1.0, the world's first open-source mobile OS. In April 2009, Android 1.6 added Google Maps. That same year Motorola released the Droid mobile device. The next significant release was Android 2.2, nicknamed Froyo (short for Frozen Yogurt), which offered an OS tune-up for speed, USB tethering for WiFi hot spots, and support for Adobe Flash 10.1 for watching videos on the built-in browser. In 2010, Motorola released the Backflip and the Droid X, and T-Mobile released the G2. In February 2011, Android released 3.0, the made-for-tablet installment. This brief history mentions just a few Android devices; today there are dozens, and the number is steadily increasing.

The Android platform, based on the Linux OS, is designed to be a general-purpose handheld computing platform. The Linux core controls the mobile device's memory, internal devices, and processes. The Android libraries control telephony, video, graphics, and the user interface. Like any

Linux system, the Android OS is a multiuser system in which each application is treated as a different user with a unique user ID. The OS sets permissions on files based on the application's ID so that applications have access to the necessary files.

The Android software development kit (SDK) supports most of the Java Standard Edition. However, Android replaces the Java abstract windowing toolkit (AWT) and Swing packages with its own user interface (UI) framework. The popularity of the Java programming language, along with the extensive class library associated with the ADK, makes it an attractive development platform. As you might expect as a Java programmer, each application runs on its own Java virtual machine (JVM). However, Android supplies its own optimized JVM called the Dalvik virtual machine.

History of android

- Android Inc. was founded in Palo Alto, California, in October 2003 by Andy Rubin, Rich Miner, Nick Sears, and Chris White.
- Rubin described the Android project as "tremendous potential in developing smarter mobile devices that are more aware of its owner's location and preferences".
- The early intentions of the company were to develop an advanced operating system for digital cameras, and this was the basis of its pitch to investors in April 2004.
- The company then decided that the market for cameras was not large enough for its goals, and by five months later it had diverted its efforts and was pitching Android as a handset operating system that would rival Symbian and Microsoft Windows Mobile.
- Rubin had difficulty attracting investors early on, and Android was facing eviction from its office space. Steve Perlman, a close friend of Rubin, brought him $10,000 in cash in an envelope, and shortly thereafter wired an undisclosed amount as seed funding.
- In July 2005, Google acquired Android Inc. for at least $50 million.
- Its key employees, including Rubin, Miner and White, joined Google as part of the acquisition.
- Speculation about Google's intention to enter the mobile communications market continued to build through December 2006.
- An early prototype had a close resemblance to a BlackBerry phone, with no touch screen and a physical QWERTY keyboard, but the arrival of 2007's Apple iPhone meant that Android "had to go back to the drawing board".[23][24] Google later changed its Android specification documents to state that "Touch screens will be supported", although "the

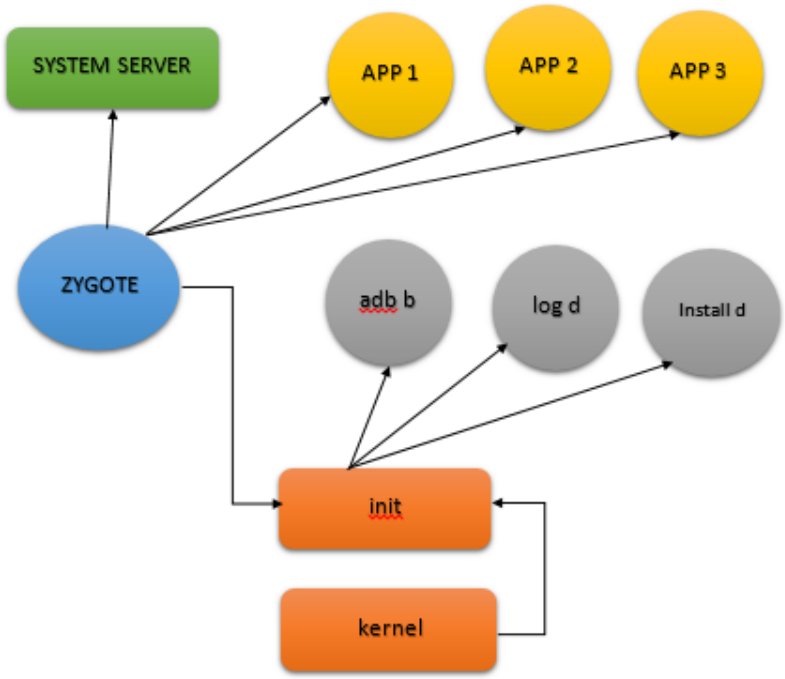
Product was designed with the presence of discrete physical buttons as an assumption, therefore a touch screen cannot completely replace physical buttons".

- By 2008, both Nokia and BlackBerry announced touch-based smart phones to rival the iPhone 3G, and Android's focus eventually switched to just touch screens. The first commercially available Smartphone running Android was the HTC Dream, also known as T-Mobile G1, announced on September 23, 2008.
- On November 5, 2007, the Open Handset Alliance, a consortium of technology companies including Google, device manufacturers such as HTC, Motorola and Samsung, wireless carriers such as Sprint and T-Mobile, and chipset makers such as Qualcomm and Texas Instruments, unveiled itself, with a goal to develop "the first truly open and comprehensive platform for mobile devices".[28][29][30] Within a year, the Open Handset Alliance faced two other open source competitors, the Symbian Foundation and the LiMo Foundation, the latter also developing a Linux-based mobile operating system like Google. In September 2007, InformationWeek covered an Evalueserve study reporting that Google had filed several patent applications in the area of mobile telephony.

ANDROID VERSIONS

NAME	INTERNAL CODENAME	VERSION NUMBER	API LEVEL	RELEASE DATE
Android 1.0	N/A	1.0	1	SEPTEMBER 23, 2008
Android 1.1	PETIT FOUR	1.1	2	FEBRUARY 09,2009
Android Cupcake	CUPCAKE	1.5	3	APRIL 27, 2009
Android Donut	DONUT	1.6	4	SEPTEMBER 15, 2009
Android Eclair	ÉCLAIR	2.0	5	OCTOBER 27, 2009
		2.0.1	6	DECEMBER 03, 2009
		2.1	7	JANUARY 11, 2010
Android Froyo	FROYO	2.2 – 2.2.3	8	MAY 20, 2010
Android Gingerbread	GINGERBREAD	2.3 – 2.3.2	9	DECEMBER 06, 2010
		2.3.3 – 2.3.7	10	FEBRUARY 09, 2011
Android Honeycomb	HONEYCOMB	3.0	11	FEBRUARY 22, 2011
		3.1	12	MAY 10, 2011
		3.2 – 3.2.6	13	JULY 15,2011
Android Ice Cream Sandwich	ICE CREAM SANDWICH	4.0 – 4.0.2	14	OCTOBER 18,2011
		4.0.3 – 4.0.4	15	DECEMBER 16, 2011
Android Jelly Bean	JELLY BEAN	4.1 – 4.1.2	16	JULY 9, 2012
		4.2 – 4.2.2	17	NOVEMBER 13, 2012
		4.3 – 4.3.1	18	JULY 24, 2013
Android KitKat	Key Lime Pie	4.4 – 4.4.4	19	October 31, 2013
		4.4W – 4.4W.2	20	June 25, 2014
Android Lollipop	Lemon Meringue Pie	5.0 – 5.0.2	21	November 4, 2014
		5.1 – 5.1.1	22	March 2, 2015[18]
Android Marshmallow	Macadamia Nut Cookie	6.0 – 6.0.1	23	October 2, 2015
Android Nougat	New York Cheesecake	7.0	24	August 22, 2016
		7.1 – 7.1.2	25	October 4, 2016
Android Oreo	Oatmeal Cookie	8.0	26	August 21, 2017
		8.1	27	December 5, 2017
Android Pie	Pistachio Ice Cream	9	28	August 6, 2018
Android 10	Quince Tart	10	29	September 3, 2019
Android 11	Red Velvet Cake	11	30	September 8, 2020
Android 12	Snow Cone	12	31	October 4, 2021
Android 12L	Snow Cone v2	**12.1**	32	March 7, 2022

ANDROID INTERNALS



1. KERNEL

- Computer program – core of computer's operating system.
- Computer controls of everything inside of the system.
- First program to be loaded on startup and it handles the rest of the startup process.
- It handle the CPU, Memory.
- Manages IO devices.
- Communicates with other processes through system calls.
- System calls occurs when other processes want to do something that is hardware related.

Modification in Original kernel

- (Linux kernel) makes Android runs on phones.
- Which have limited memory.
- Not always connected to a power source.

Out of Memory killer

- This helps to kill apps that users leave running.

Wakelocks

- Allows Android to sleep and consumes as little energy as possible.

2. INIT

- The init process is picked up by the kernel run on startup.
- Root of all other processes, which will be created from this process, spawning everything else.
- List of instruction and setting about other processes and how they need to be configured.
- Most of the things that init will start up are called daemons (background process).
- adbd (Android Debug Bridge Daemon) or installed that's being used to install apps.

3. ZYGOTE

- Special process (special daemon) in Android which handles the forking of each new application process.
- These processes are simply regular linux processes.
- Zygote – the template process for each app and services that is started on the devices.
- It is launched by the Android runtime, which also starts the first virtual machine (VM).
- The VM then calls Zygote's main () methods which causes Zygote to preload all shared Java classes and resources into memory.

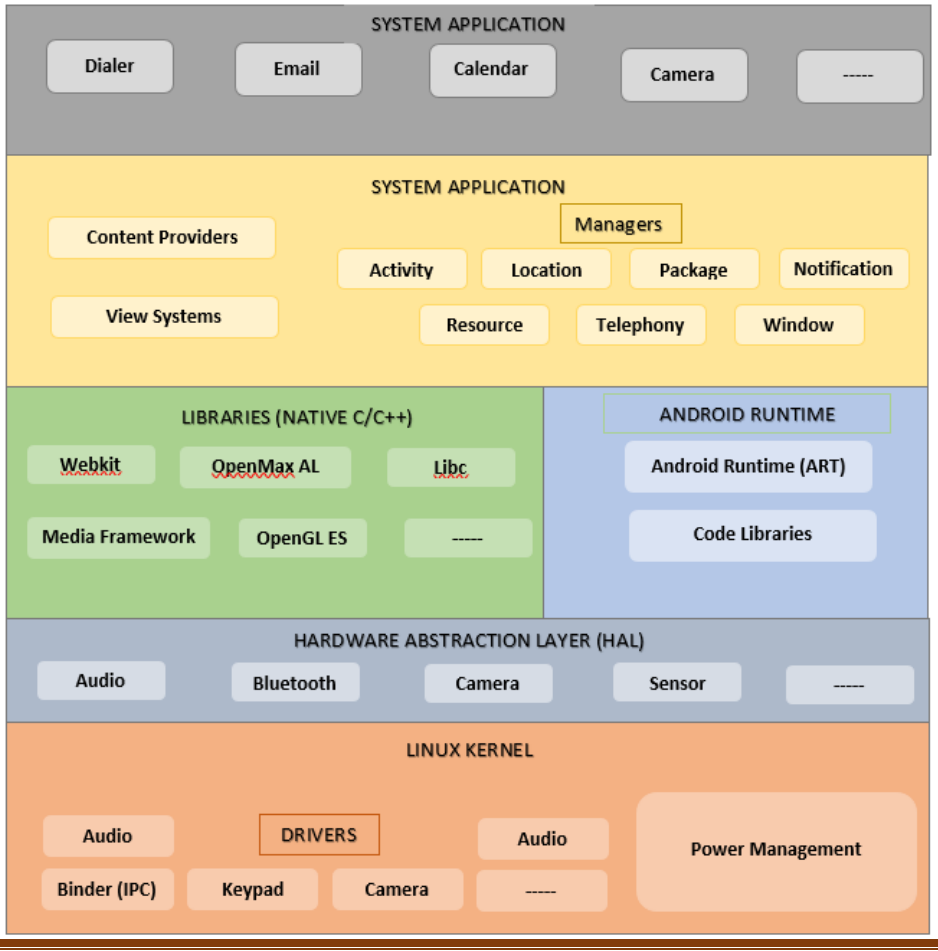
4. BINDER

- Binder allows inter – app communication is a safe way.
- Our app is not directly talking to the other app.

5. SYSTEM SERVER

- Heart of Android and it's responsible for everything Android does.
- The Windows Manager is responsible for windowing and being in-charge all things in window.
- It keeps tracks of window activities and it keeps track of what is visible.
- It is also responsible for transitions between activities for an overall good UI experience.
- The Package manager is responsible for installing new applications.

ANDROID ARCHITECTURE



- Android Architecture (or) Android Software Stack.
- Linux Kernel
- Hardware Abstraction Layer
- Native Libraries (Middleware)
- Android Runtime
- Application Framework
- System Application

1. Linux Kernel

- This layer is the foundation of the Android platform.
- Contains all low level drivers for various hardware components support.
- Android Runtime relies on Linux kernel for core system services like,
    - Memory, process management, threading etc.
    - Network stack
    - Driver model
    - Security and more.

2. Hardware Abstraction Layer

- Abstraction between hardware and rest of the software stack.
- Provides standard interfaces that expose device hardware capabilities to the higher level Java API.
- It consists of multiple library modules.
- Each of which implements an interface for a specific type of hardware components.
- Components such as camera, Bluetooth.

3. Android Runtime

- Each app runs in its own process and with its own instance of the Android Runtime (ART) virtual machine.
- This makes process management more crucial.
- ART uses DEX files which is a type of bytecode, specially designed for Android, which helps ART to manage memory more efficiency.
- Contains set of core libraries that enables developers to write Android Apps using Java Programming.
- Prior to version 5.0 – Dalvik was used as the run time along with JIT compiler.

Just In Time (JIT)

- With the Dalvik JIT compiler, each time when the app is run, it dynamically translates a part of the Dalvik bytecode into machine code.
- As the execution progresses, more bytecode is compiled and cached.
- Since JIT compiles only a part of the code, it has a smaller memory footprint and less physical space on the devices.

Ahead of Time (AOT)

- ART is equipped with an Ahead-of-Time compiler.
- During the app's installation phase, it statically translates the DEX bytecode into machine code and stores in the device's storage.
- This is a one-time event which happens when the app is installed on the device.
- With no need for JIT compilation, the code executes much faster.

4. Libraries

- Many core Android System components and services such as ART and HAL are built from native code that require native libraries written in C and C++.
- The Android platform provides Java framework APIs to expose the functionality of some of these native libraries to apps.
- If you are developing an app that requires C or C++ code, you can use the Android NDK to access some of these native platform libraries directly from your native code.

5. Java API Framework

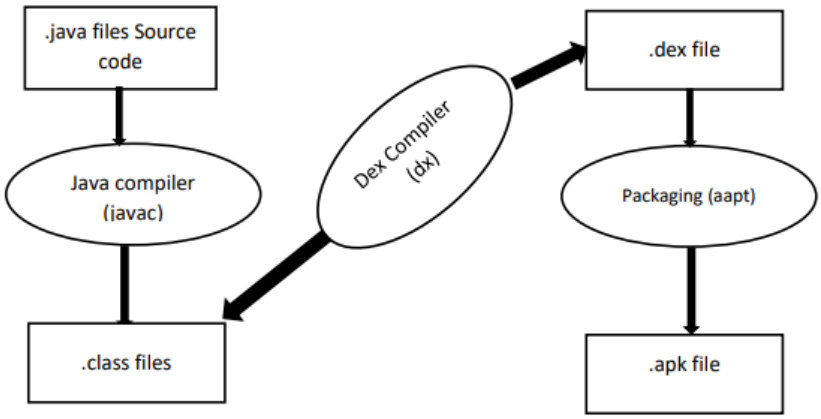
- Window Manager
  - Manages windows and drawing surfaces, and is an abstraction of the surface manager library.
- Package Manager
  - Manages various kinds of information related to the application package that are currently installed on the device.
- Telephony Manager
  - Enables app to use phone capabilities of the device.
- Location Manager
  - Deals with location awareness capabilities

5. System Apps

- Android comes with a set of core apps for email, SMS messaging, calendars, internet browsing, contacts, and more.
- Apps includes with the platform have no special status among the apps users chooses to install.
- So a third-party app can become the user's default web browser, SMS messenger, or even the default keyboard (some exceptions apply, such as the system's setting app).

Dalvik Execution

- Programs are commonly written in Java and compiled to bytecode.
- It is then translated to Dalvik bytecode.
- .dex (Dalvik Executable) and .odex (optimized Dalvik Executable)
- The compact Dalvik Executable format is designed for systems that are constrained in terms of memory and processor speed.



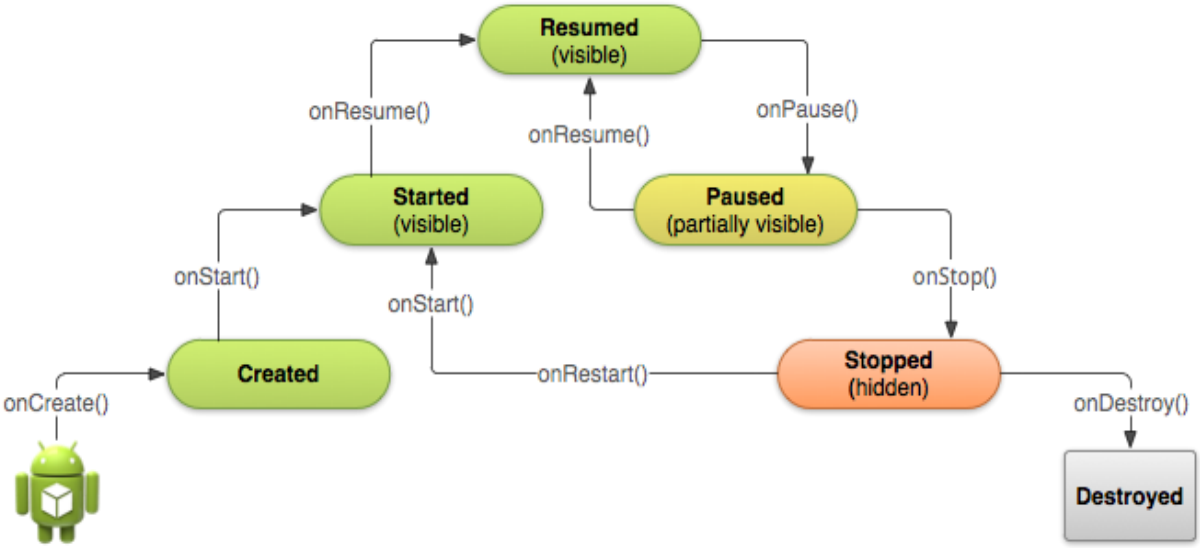
- A tool called dx is used to convert Java .class files into the .dex format.
- Multiple classes are included in a single .dex file.
- Duplicate strings and other constants used in multiple class files are included only once in the .dex output to conserve space.
- An uncmpressed .dex file is typically a few percent smaller in size than a compressed Java archive (JAR) derived from the same .class files.

How android code execution works?

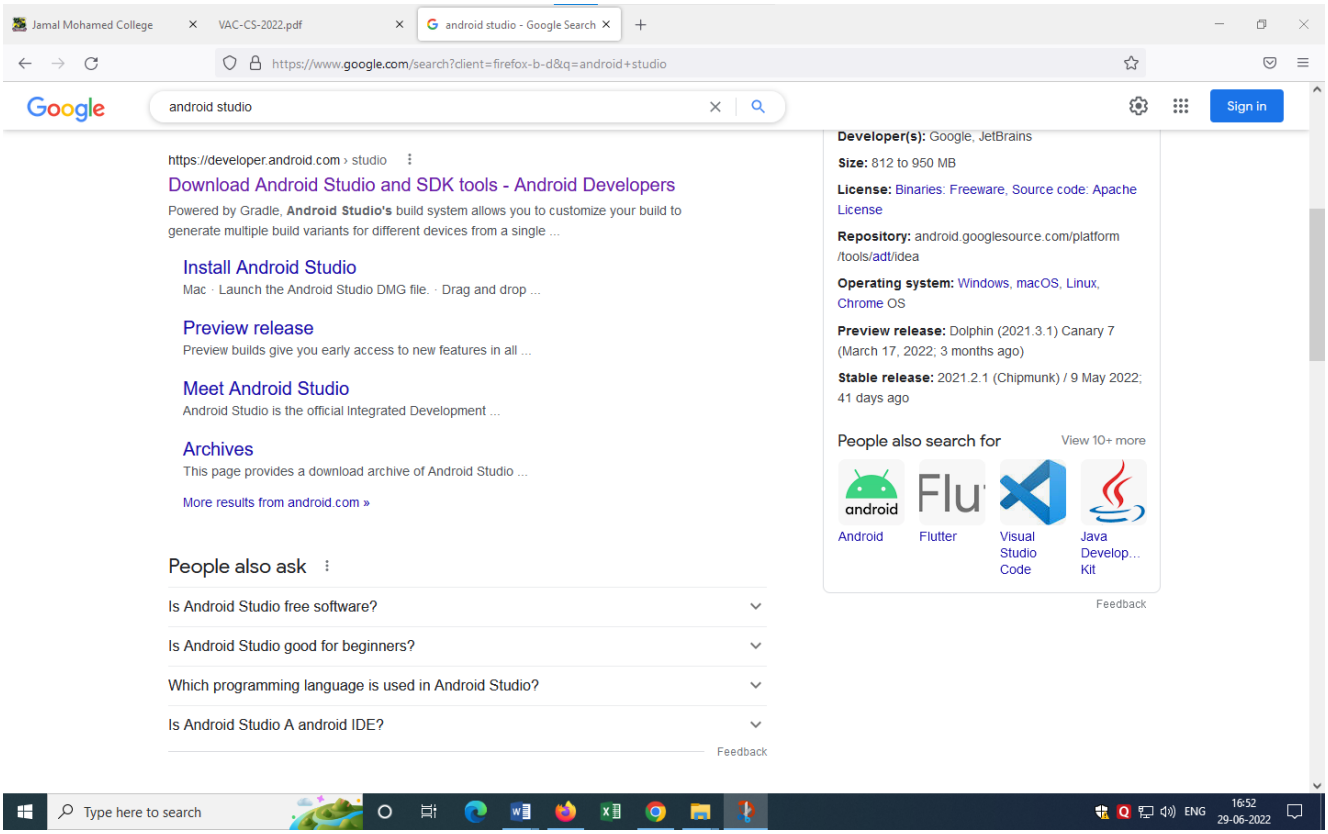
- In Android Java classes converted into DEX bytecode.
- The DEX bytecode format is translated to native machine code via either ART or the Dalvik runtimes.
- Here DEX bytecode is independent of device architecture.
- Dalvik is a JIT (just in time) compilation based engine.
- There were drawbacks to use Dalvik hence from Android 4.4 (kitkat) ART was introduced as runtime and from Android 5.0 (Lollipop).
- It has completely replace Dalvik.
- Android 7.0 adds a just in time (JIT) compiler with code profiling to Android runtime (ART) that constantly improves the performance of Android apps as they run.
- Dalvik uses JIT (just in time) compilation whereas ART uses AOT (Ahead of Time) compilation.

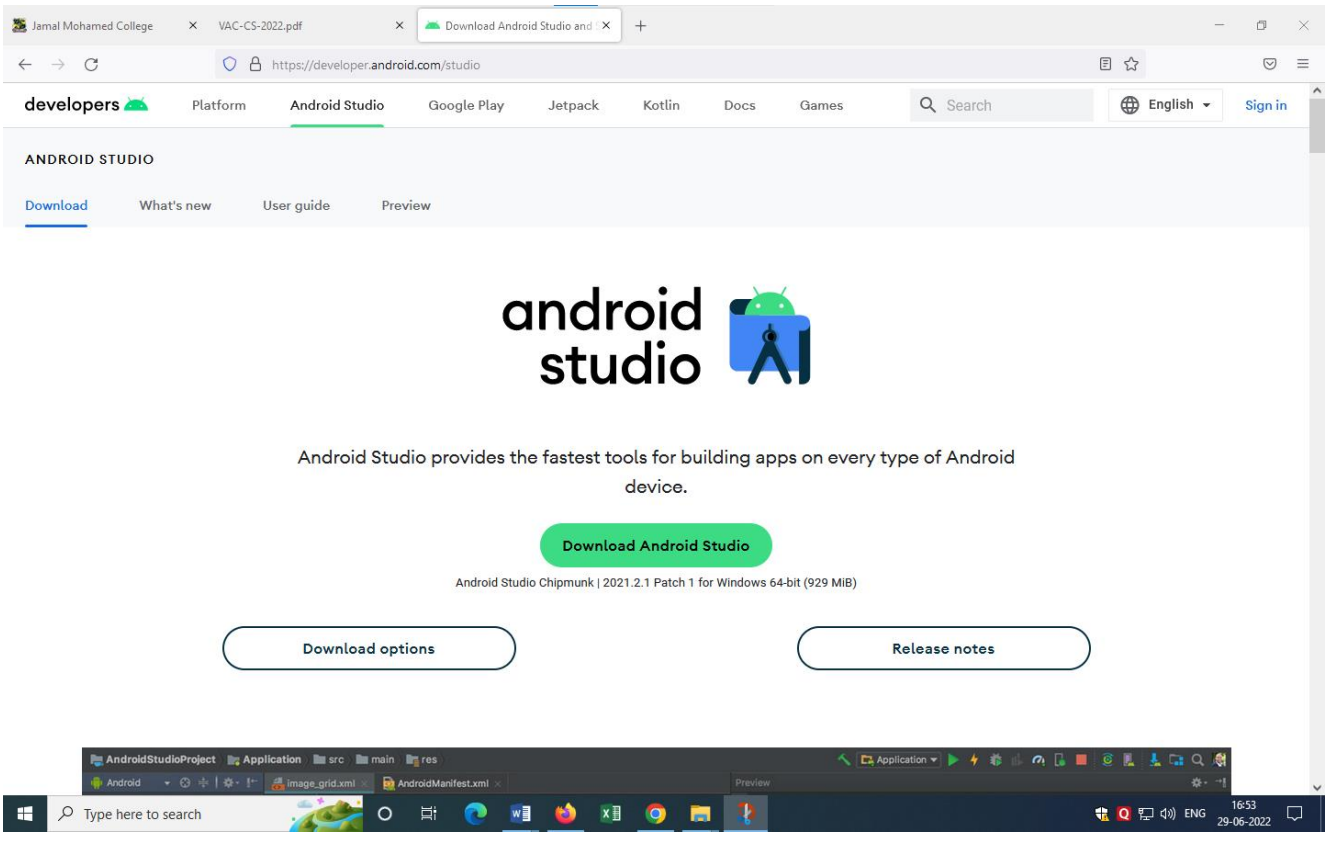
ANDROID LIFE CYCLE

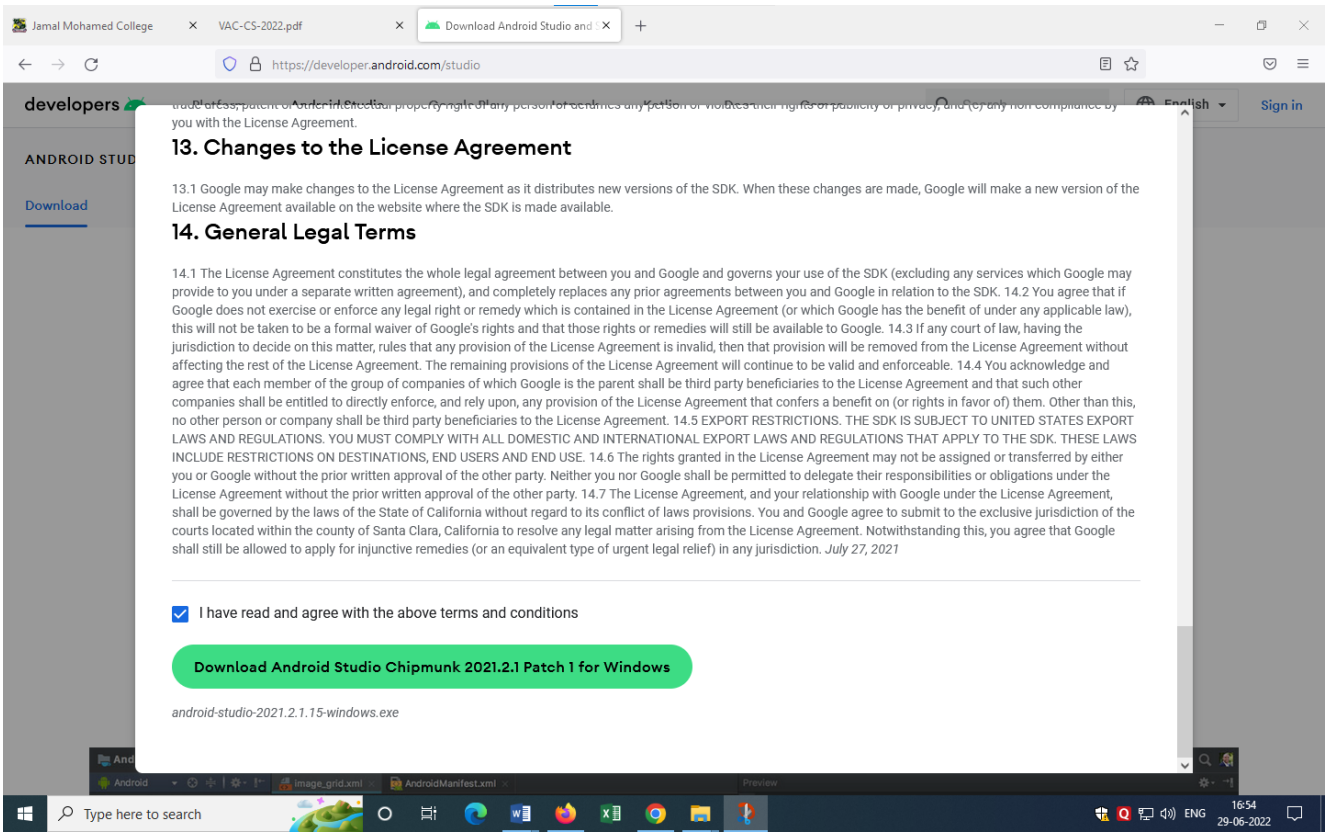
Method	Description
onCreate	called when activity is first created.
onStart	called when activity is becoming visible to the user.
onResume	called when activity will start interacting with the user.
onPause	called when activity is not visible to the user.
onStop	called when activity is no longer visible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

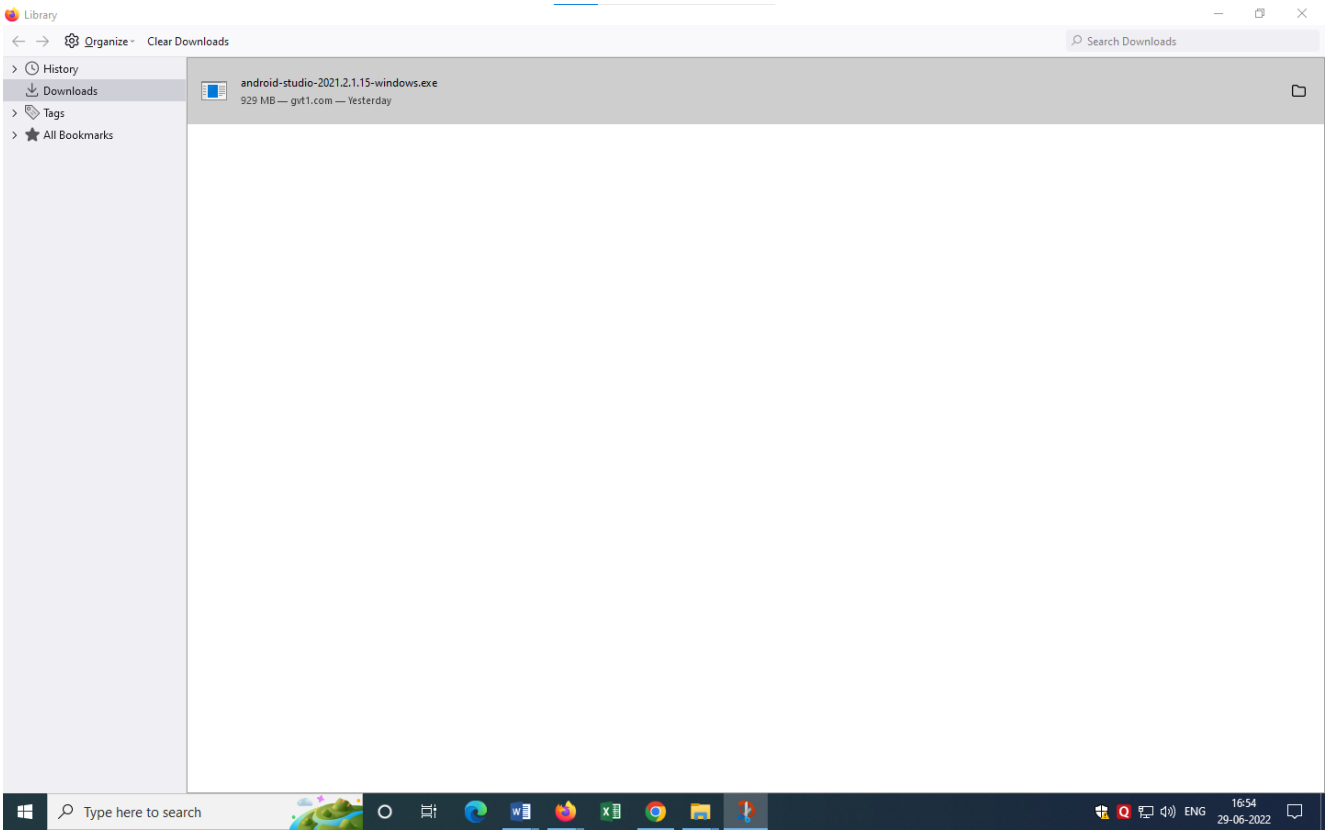

Resumed
(visible)
onResume()
onResume()
onPause()
Started
(visible)
Paused
(partially visible)
onStart()
onStart()
onStop()
Created
onRestart()
Stopped
(hidden)
onCreate()
onDestroy()
Destroyed

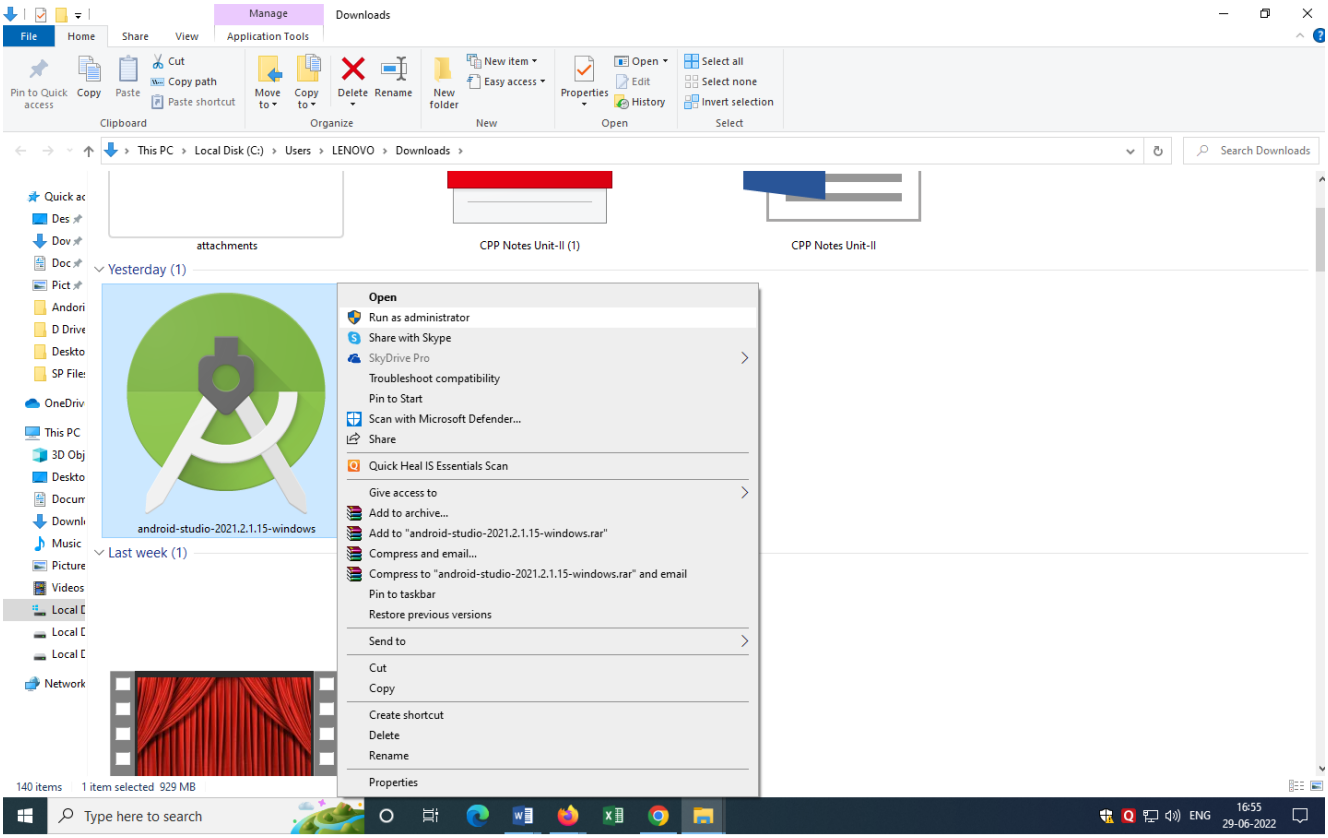
ANDROID STUDIO DOWNLOAD & INSTALLATION

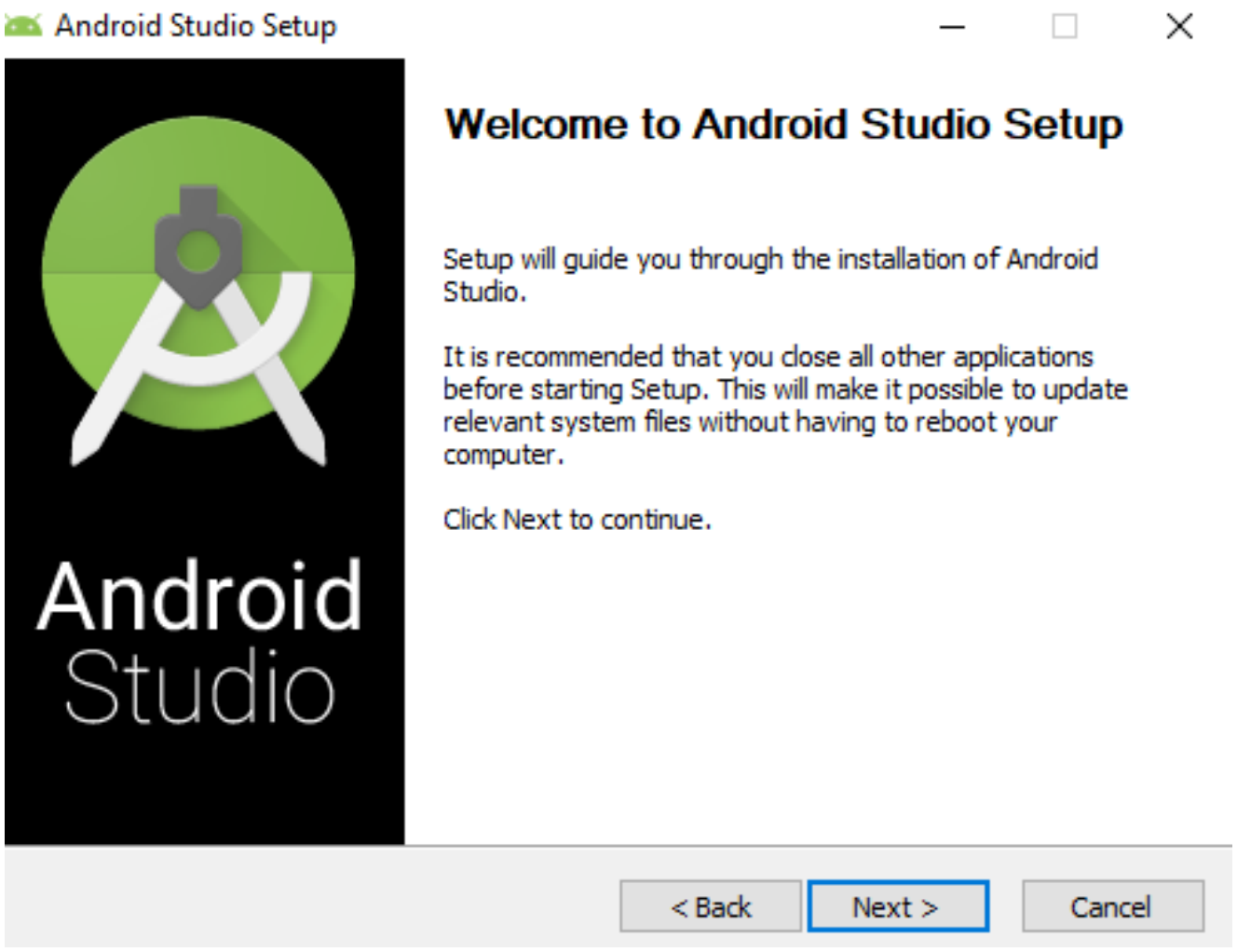


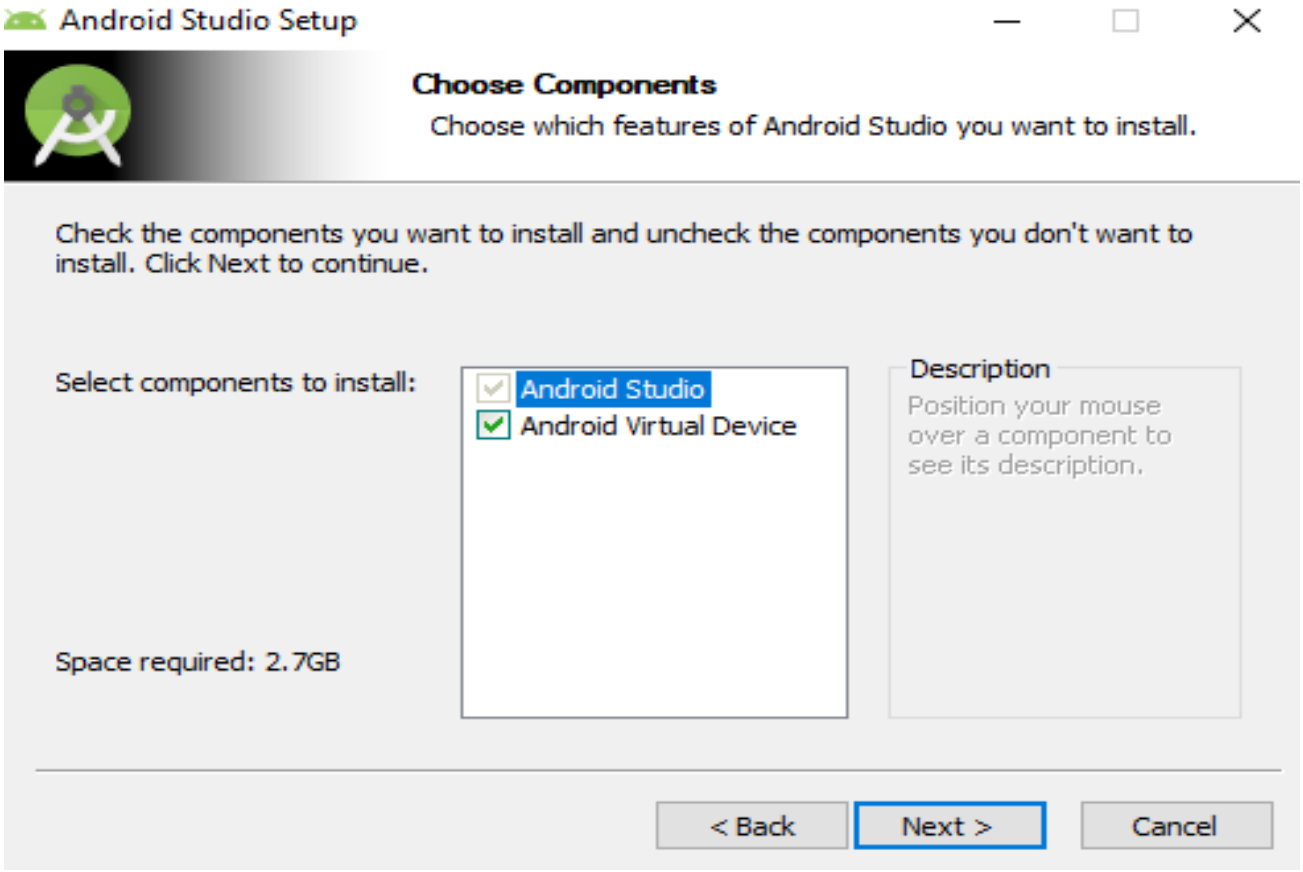


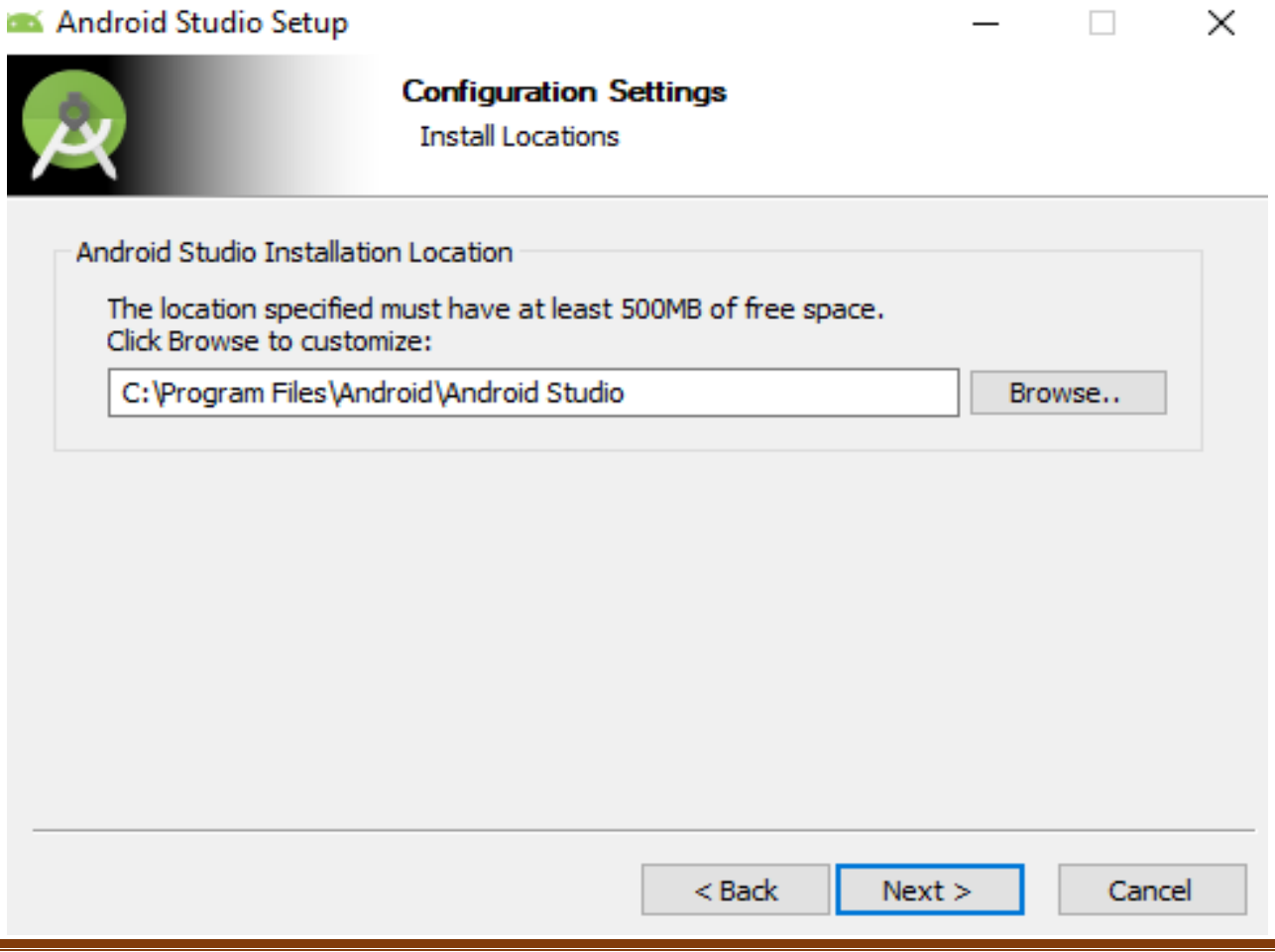

Jamal Mohamed College
VAC-CS-2022.pdf
Download Android Studio and
https://developer.android.com/studio
developers
Sign in
ANDROID STUD
Download
you with the License Agreement.
13. Changes to the License Agreement
13.1 Google may make changes to the License Agreement as it distributes new versions of the SDK. When these changes are made, Google will make a new version of the License Agreement available on the website where the SDK is made available.
14. General Legal Terms
14.1 The License Agreement constitutes the whole legal agreement between you and Google and governs your use of the SDK (excluding any services which Google may provide to you under a separate written agreement), and completely replaces any prior agreements between you and Google in relation to the SDK. 14.2 You agree that if Google does not exercise or enforce any legal right or remedy which is contained in the License Agreement (or which Google has the benefit of under any applicable law), this will not be taken to be a formal waiver of Google's rights and that those rights or remedies will still be available to Google. 14.3 If any court of law, having the jurisdiction to decide on this matter, rules that any provision of the License Agreement is invalid, then that provision will be removed from the License Agreement without affecting the rest of the License Agreement. The remaining provisions of the License Agreement will continue to be valid and enforceable. 14.4 You acknowledge and agree that each member of the group of companies of which Google is the parent shall be third party beneficiaries to the License Agreement and that such other companies shall be entitled to directly enforce, and rely upon, any provision of the License Agreement that confers a benefit on (or rights in favor of) them. Other than this, no other person or company shall be third party beneficiaries to the License Agreement. 14.5 EXPORT RESTRICTIONS. THE SDK IS SUBJECT TO UNITED STATES EXPORT LAWS AND REGULATIONS. YOU MUST COMPLY WITH ALL DOMESTIC AND INTERNATIONAL EXPORT LAWS AND REGULATIONS THAT APPLY TO THE SDK. THESE LAWS INCLUDE RESTRICTIONS ON DESTINATIONS, END USERS AND END USE. 14.6 The rights granted in the License Agreement may not be assigned or transferred by either you or Google without the prior written approval of the other party. Neither you nor Google shall be permitted to delegate their responsibilities or obligations under the License Agreement without the prior written approval of the other party. 14.7 The License Agreement, and your relationship with Google under the License Agreement, shall be governed by the laws of the State of California without regard to its conflict of laws provisions. You and Google agree to submit to the exclusive jurisdiction of the courts located within the county of Santa Clara, California to resolve any legal matter arising from the License Agreement. Notwithstanding this, you agree that Google shall still be allowed to apply for injunctive remedies (or an equivalent type of urgent legal relief) in any jurisdiction. July 27, 2021
I have read and agree with the above terms and conditions
Download Android Studio Chipmunk 2021.2.1 Patch 1 for Windows
android-studio-2021.2.1.15-windows.exe
Type here to search
ENG
16:54
29-06-2022

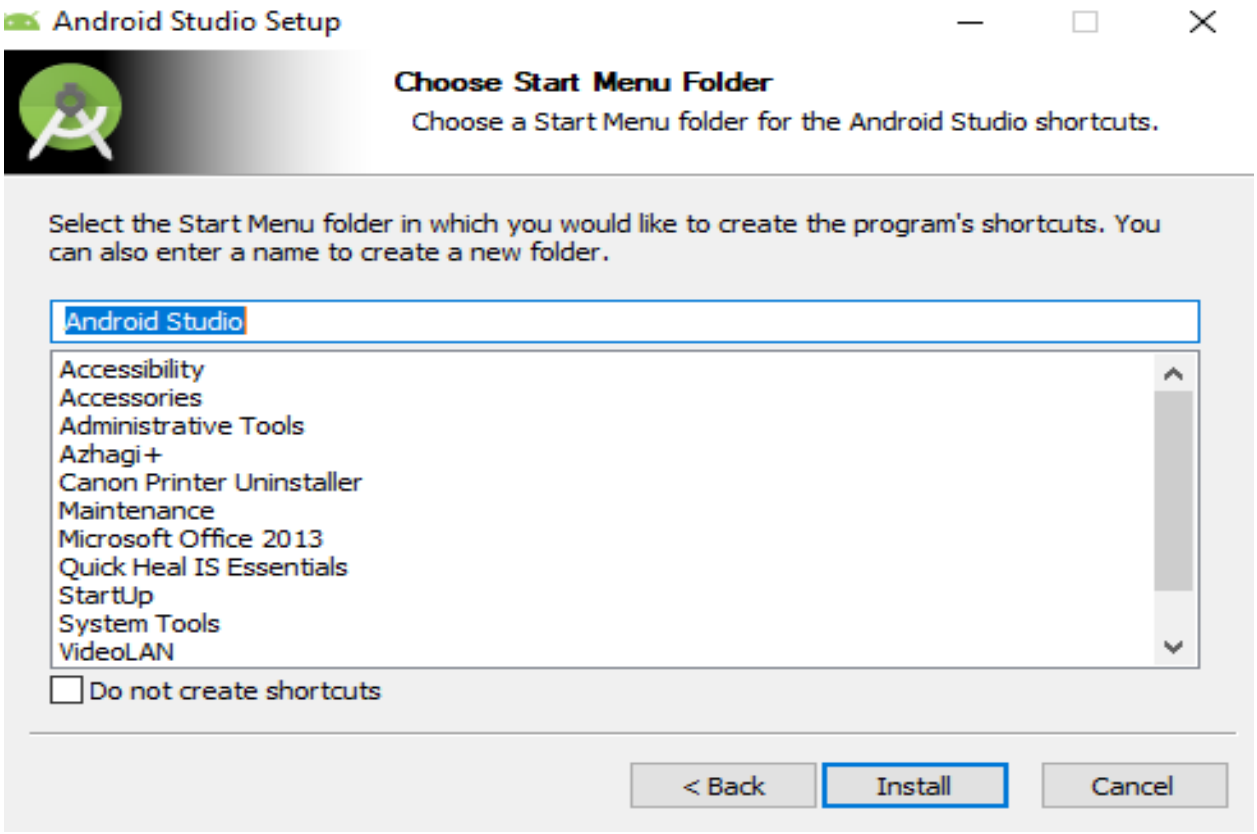

Library
Organize
Clear Downloads
Search Downloads
History
Downloads
Tags
All Bookmarks
android-studio-2021.2.1.15-windows.exe
929 MB — gvt1.com — Yesterday
Type here to search
ENG
16:54
29-06-2022

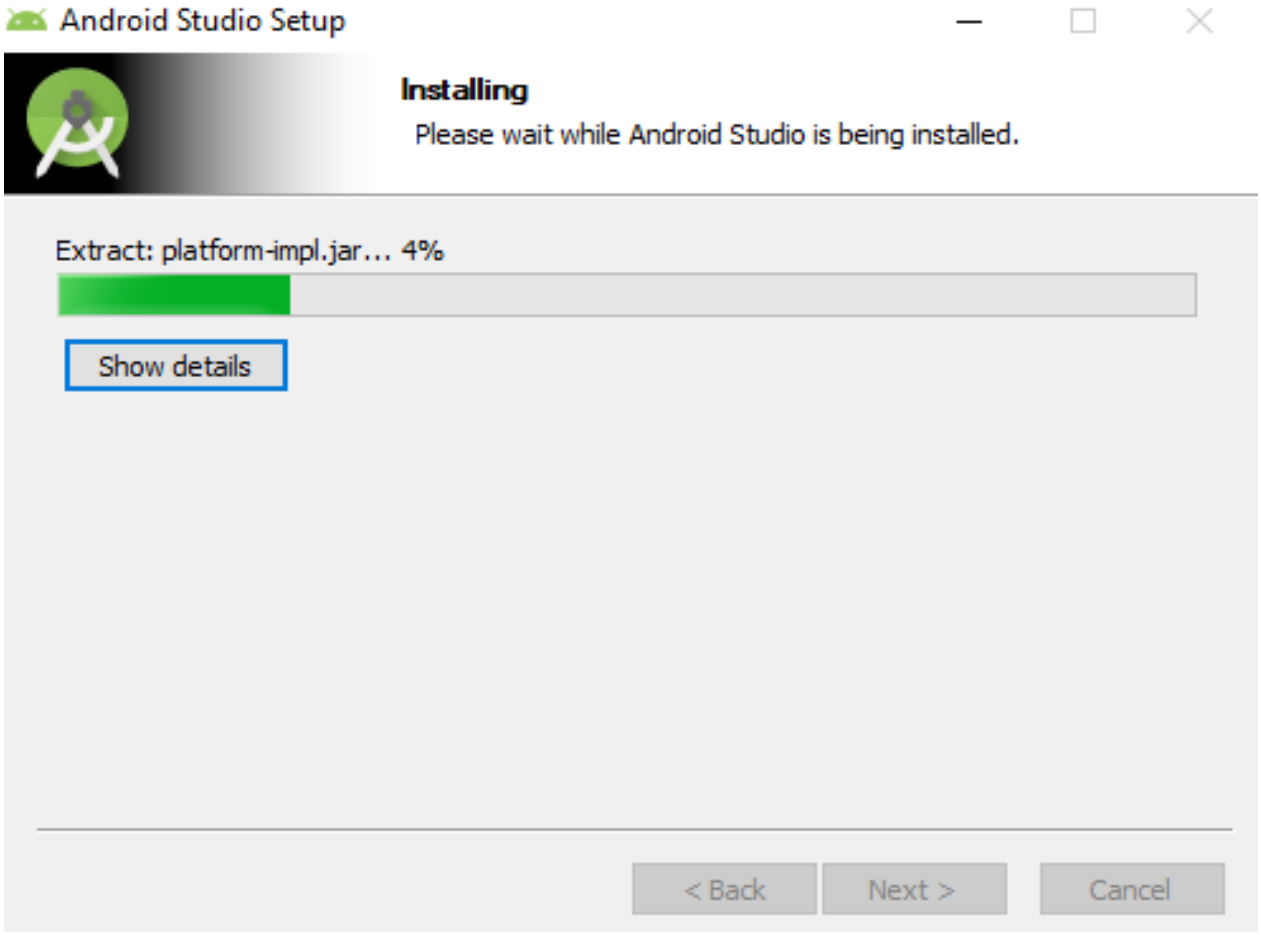

Manage
Downloads
File
Home
Share
View
Application Tools
Cut
Copy path
Paste shortcut
Pin to Quick access
Copy
Paste
Move to
Copy to
Delete
Rename
New folder
New item
Easy access
Properties
Open
Edit
History
Select all
Select none
Invert selection
Clipboard
Organize
New
Open
Select
This PC › Local Disk (C:) › Users › LENOVO › Downloads ›
Search Downloads
attachments
CPP Notes Unit-II (1)
CPP Notes Unit-II
Yesterday (1)
android-studio-2021.2.1.15-windows
Last week (1)
Open
Run as administrator
Share with Skype
SkyDrive Pro
Troubleshoot compatibility
Pin to Start
Scan with Microsoft Defender...
Share
Quick Heal IS Essentials Scan
Give access to
Add to archive...
Add to "android-studio-2021.2.1.15-windows.rar"
Compress and email...
Compress to "android-studio-2021.2.1.15-windows.rar" and email
Pin to taskbar
Restore previous versions
Send to
Cut
Copy
Create shortcut
Delete
Rename
Properties
140 items
1 item selected 929 MB
Type here to search
ENG
16:55
29-06-2022

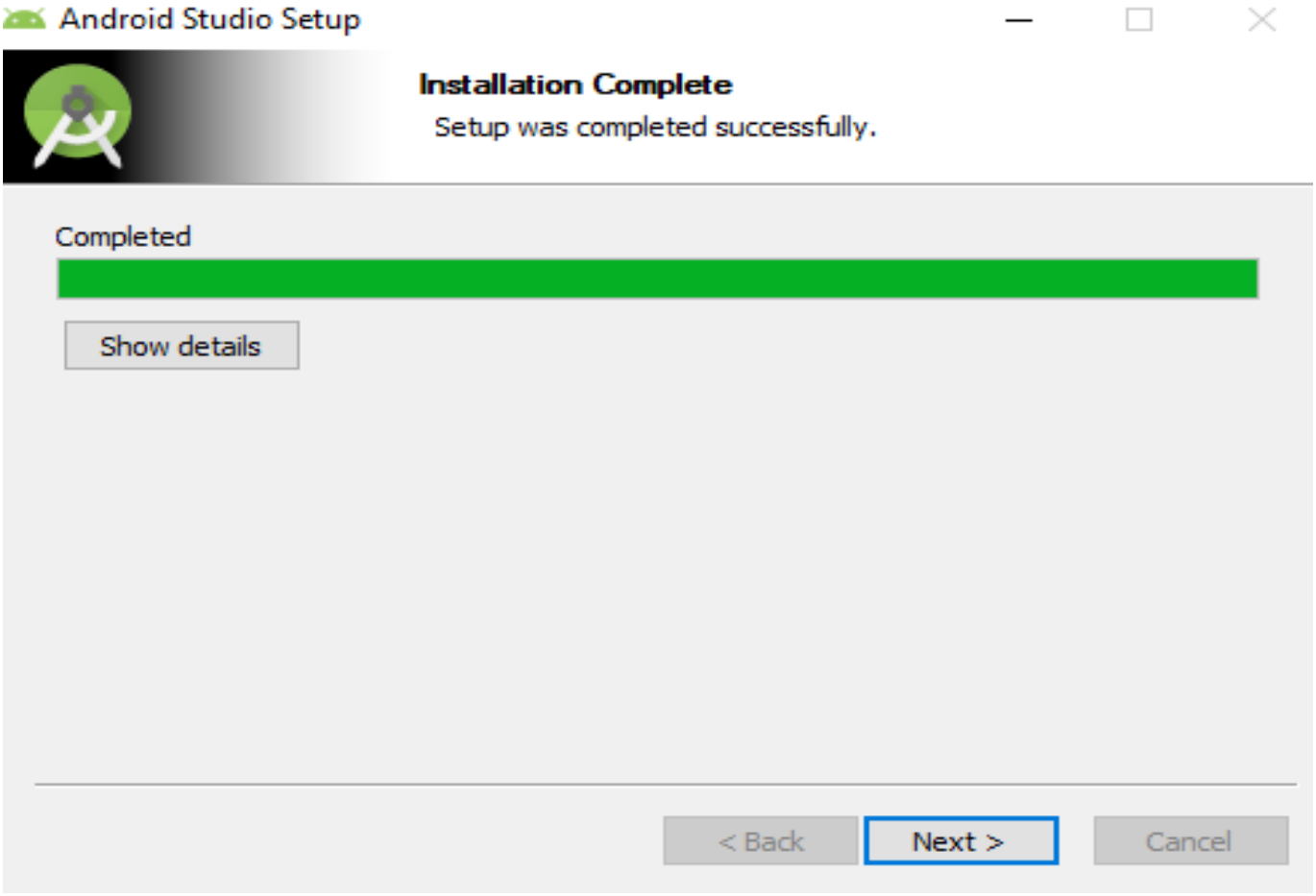

Android Studio Setup
Welcome to Android Studio Setup
Setup will guide you through the installation of Android Studio.
It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.
Click Next to continue.
Android Studio
< Back
Next >
Cancel

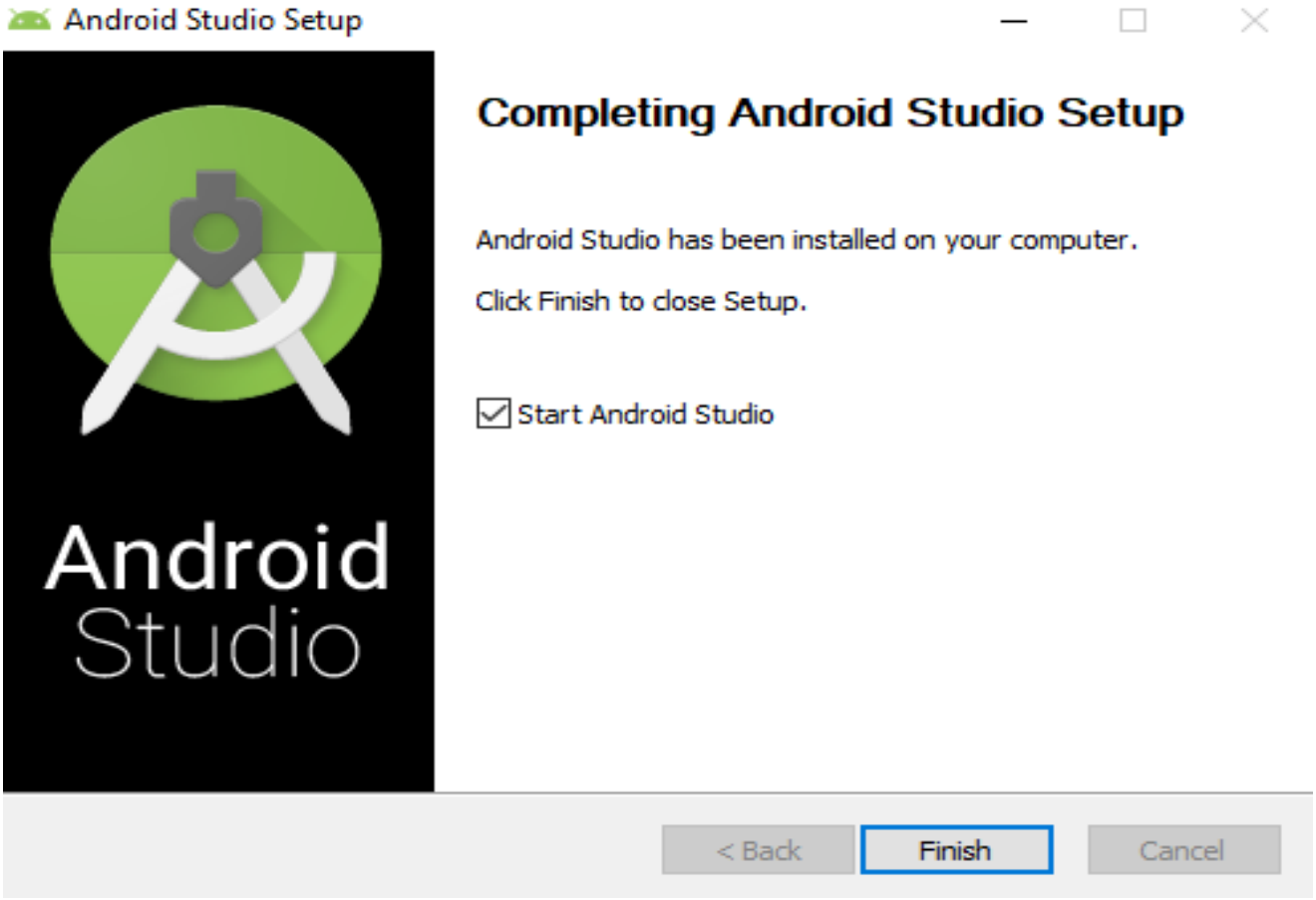

Android Studio Setup
Choose Components
Choose which features of Android Studio you want to install.
Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.
Select components to install:
Android Studio
Android Virtual Device
Description
Position your mouse over a component to see its description.
Space required: 2.7GB
< Back
Next >
Cancel

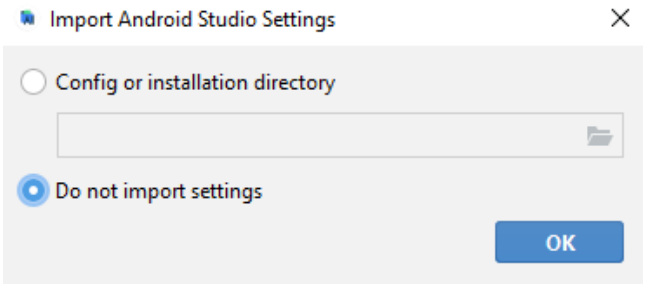

Android Studio Setup
Configuration Settings
Install Locations
Android Studio Installation Location
The location specified must have at least 500MB of free space.
Click Browse to customize:
C:\Program Files\Android\Android Studio
Browse..
< Back
Next >
Cancel

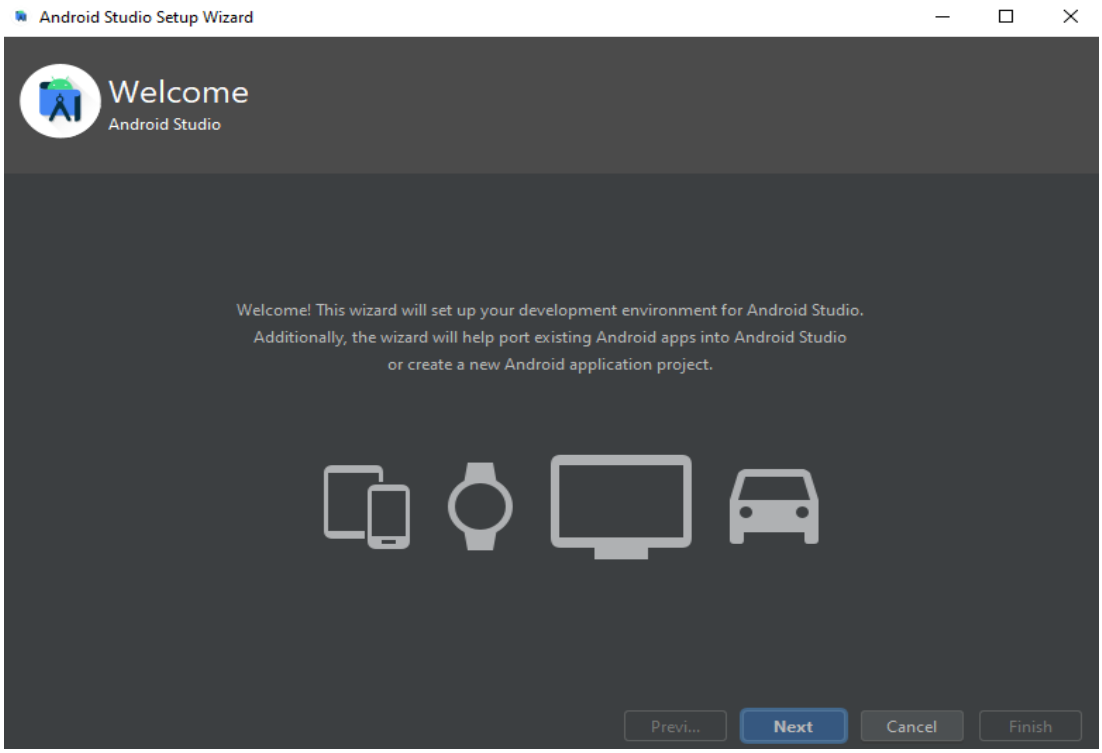

Android Studio Setup
Choose Start Menu Folder
Choose a Start Menu folder for the Android Studio shortcuts.
Select the Start Menu folder in which you would like to create the program's shortcuts. You can also enter a name to create a new folder.
Android Studio
Accessibility
Accessories
Administrative Tools
Azhagi+
Canon Printer Uninstaller
Maintenance
Microsoft Office 2013
Quick Heal IS Essentials
StartUp
System Tools
VideoLAN
Do not create shortcuts
< Back
Install
Cancel

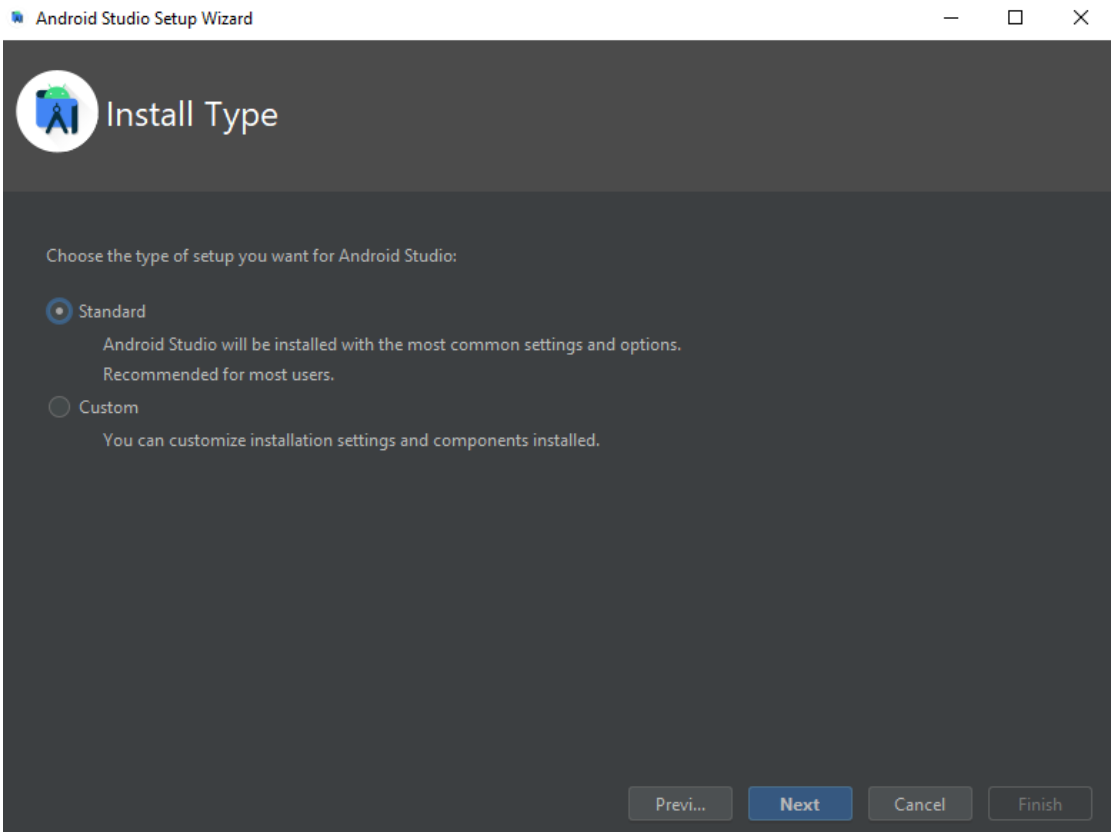

Android Studio Setup
Installing
Please wait while Android Studio is being installed.
Extract: platform-impl.jar... 4%
Show details
< Back
Next >
Cancel

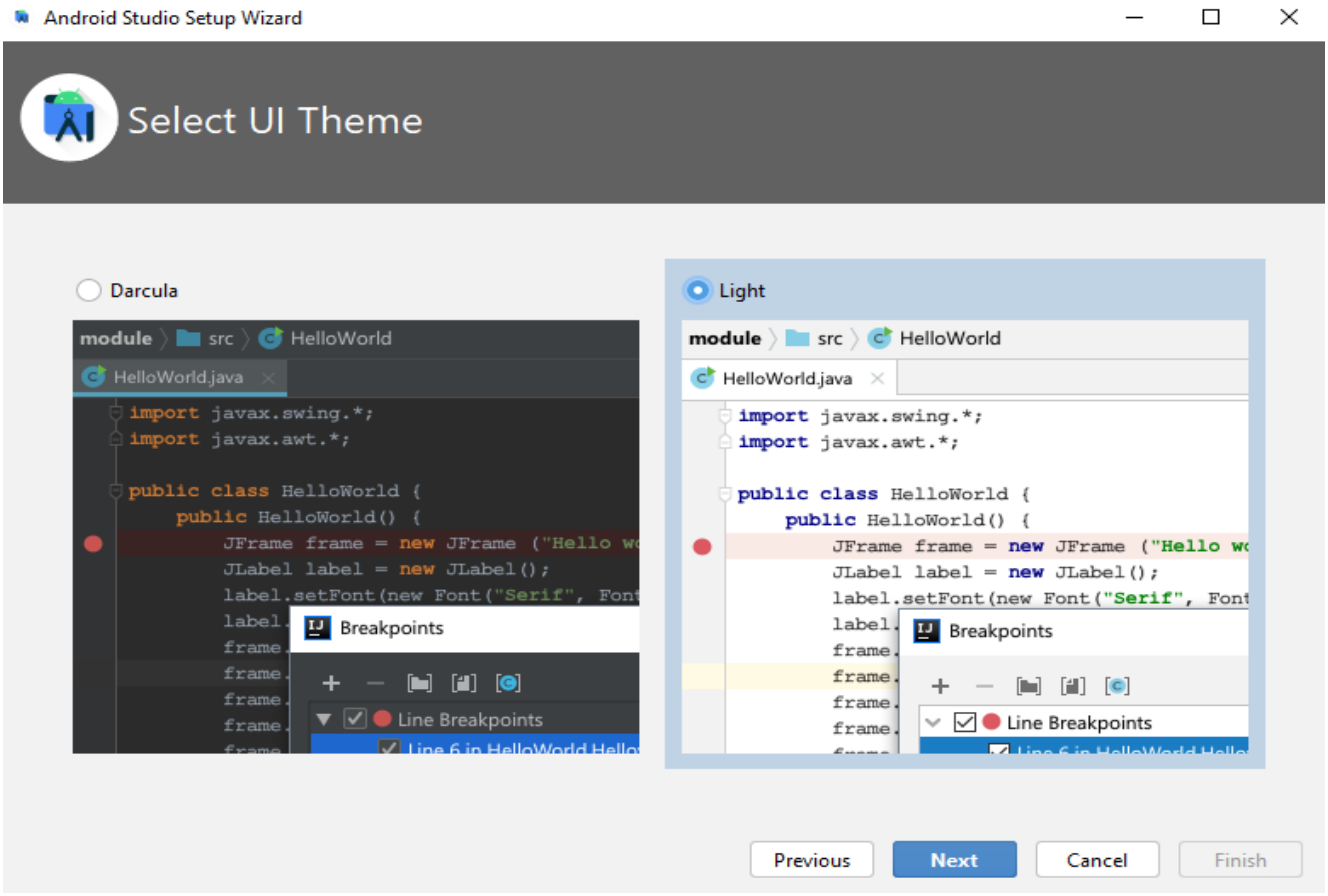

Android Studio Setup
Installation Complete
Setup was completed successfully.
Completed
Show details
< Back
Next >
Cancel

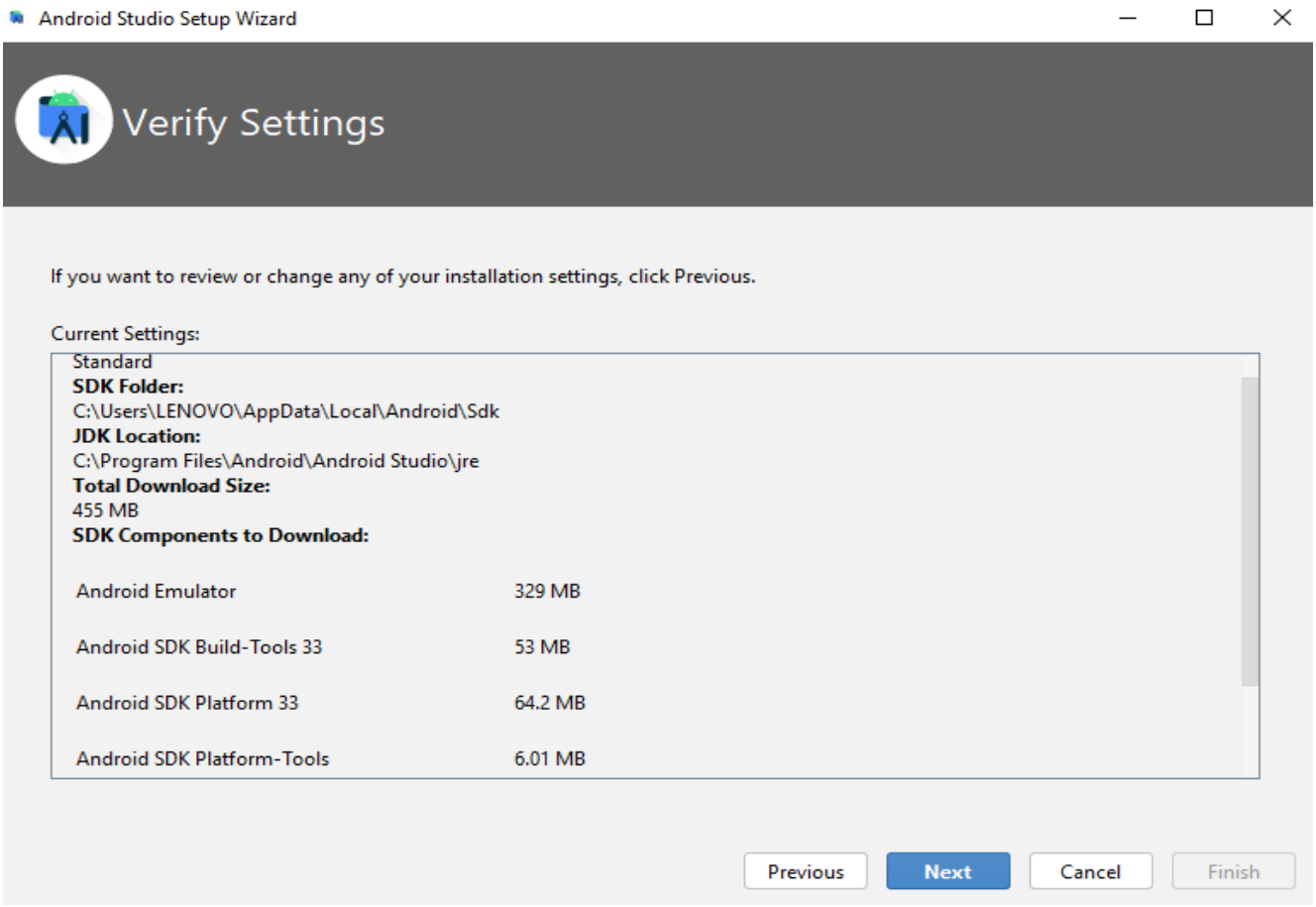

Android Studio Setup
Android
Studio
Completing Android Studio Setup
Android Studio has been installed on your computer.
Click Finish to close Setup.
Start Android Studio
< Back
Finish
Cancel

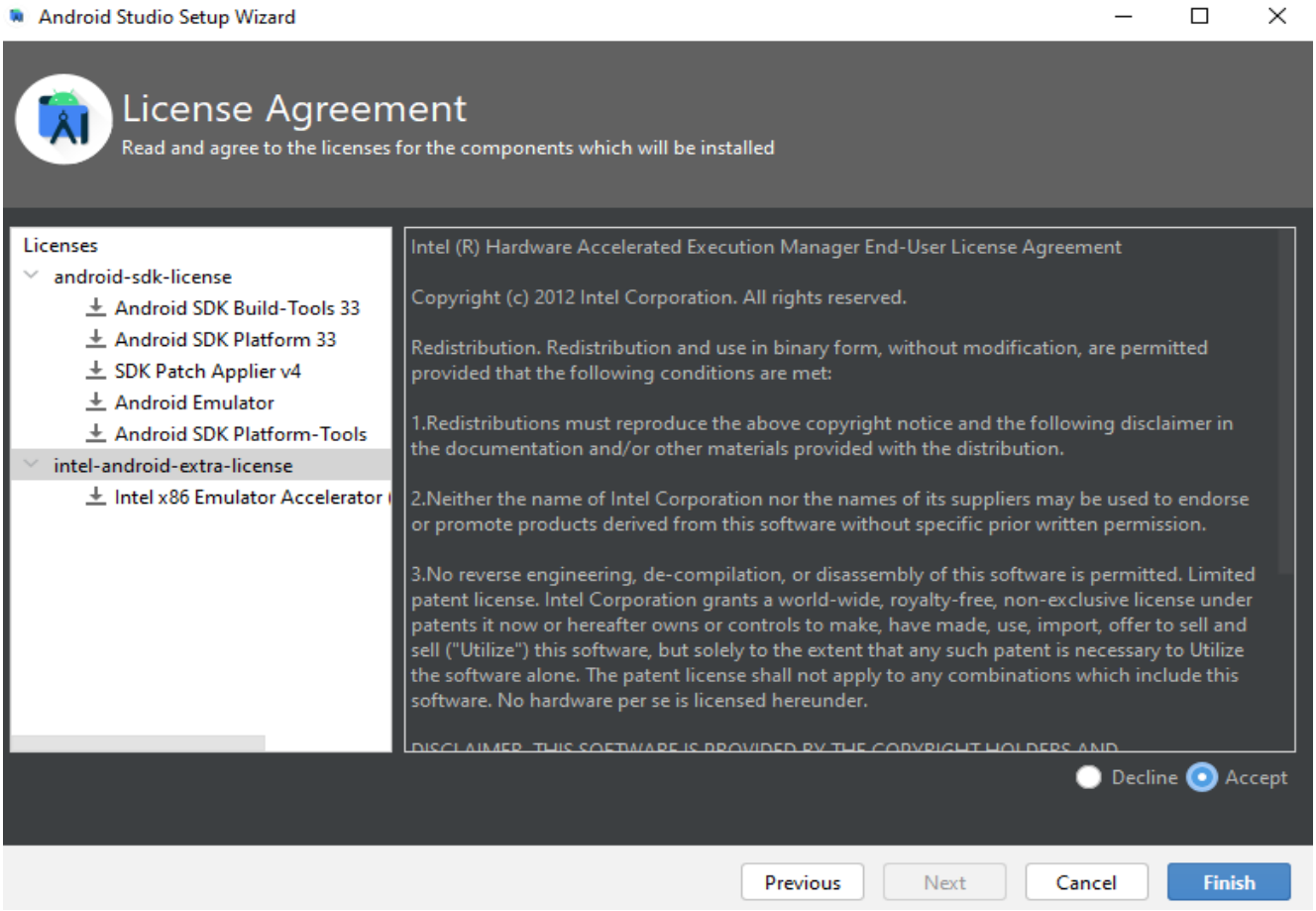

Import Android Studio Settings
Config or installation directory
Do not import settings
OK

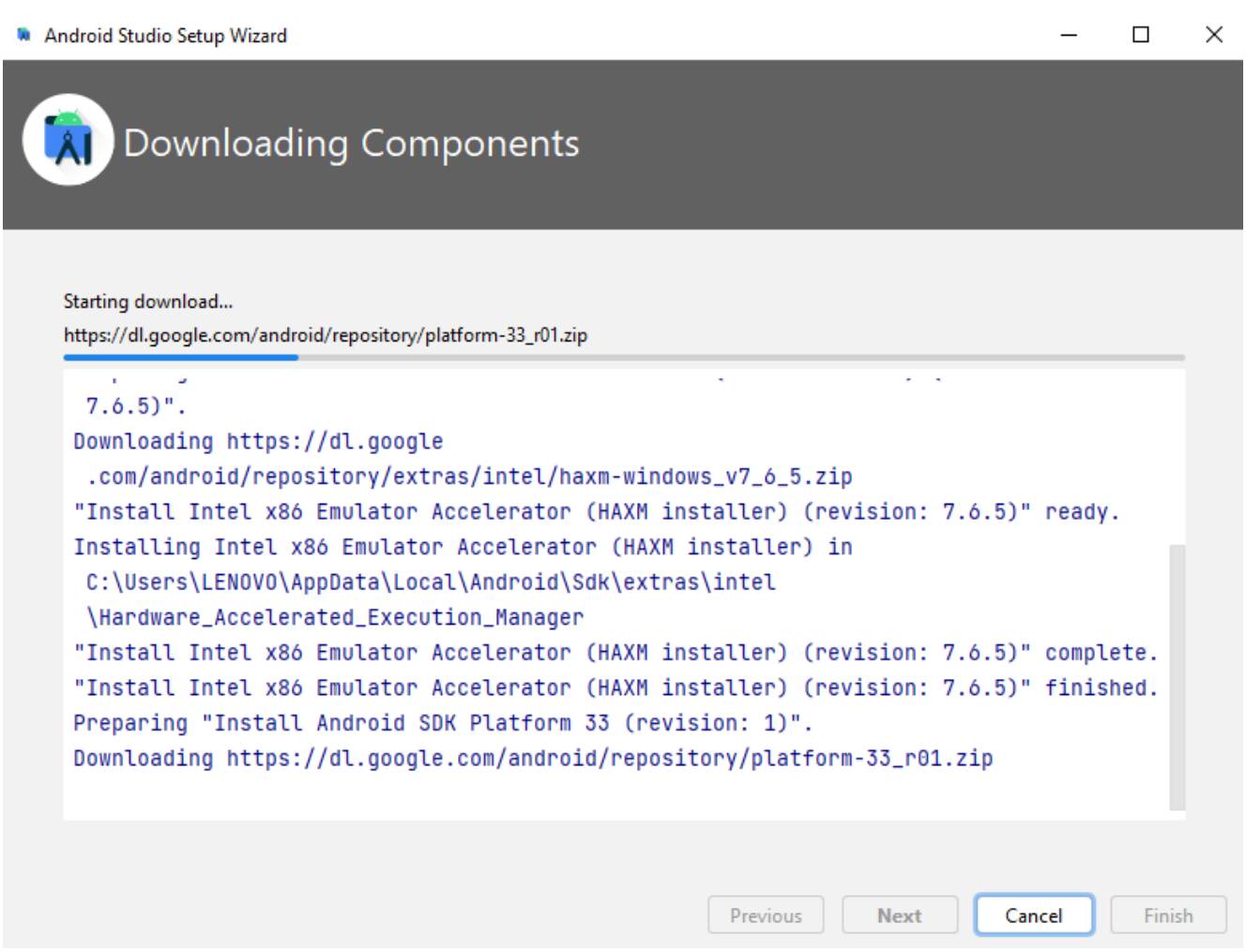

Android Studio Setup Wizard
Welcome
Android Studio
Welcome! This wizard will set up your development environment for Android Studio.
Additionally, the wizard will help port existing Android apps into Android Studio
or create a new Android application project.
Previ...
Next
Cancel
Finish

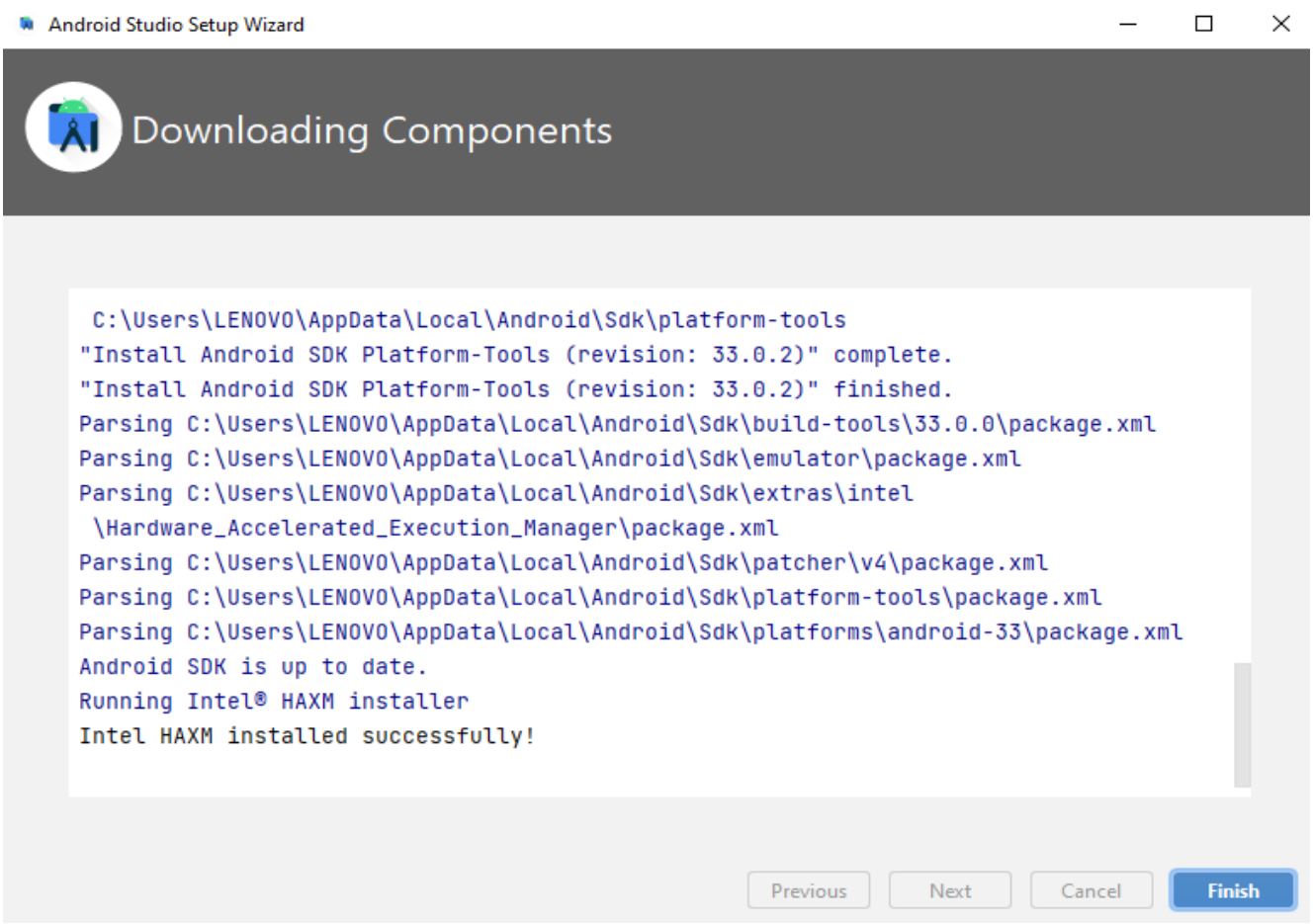

Android Studio Setup Wizard
Install Type
Choose the type of setup you want for Android Studio:
Standard
Android Studio will be installed with the most common settings and options.
Recommended for most users.
Custom
You can customize installation settings and components installed.
Previ...
Next
Cancel
Finish

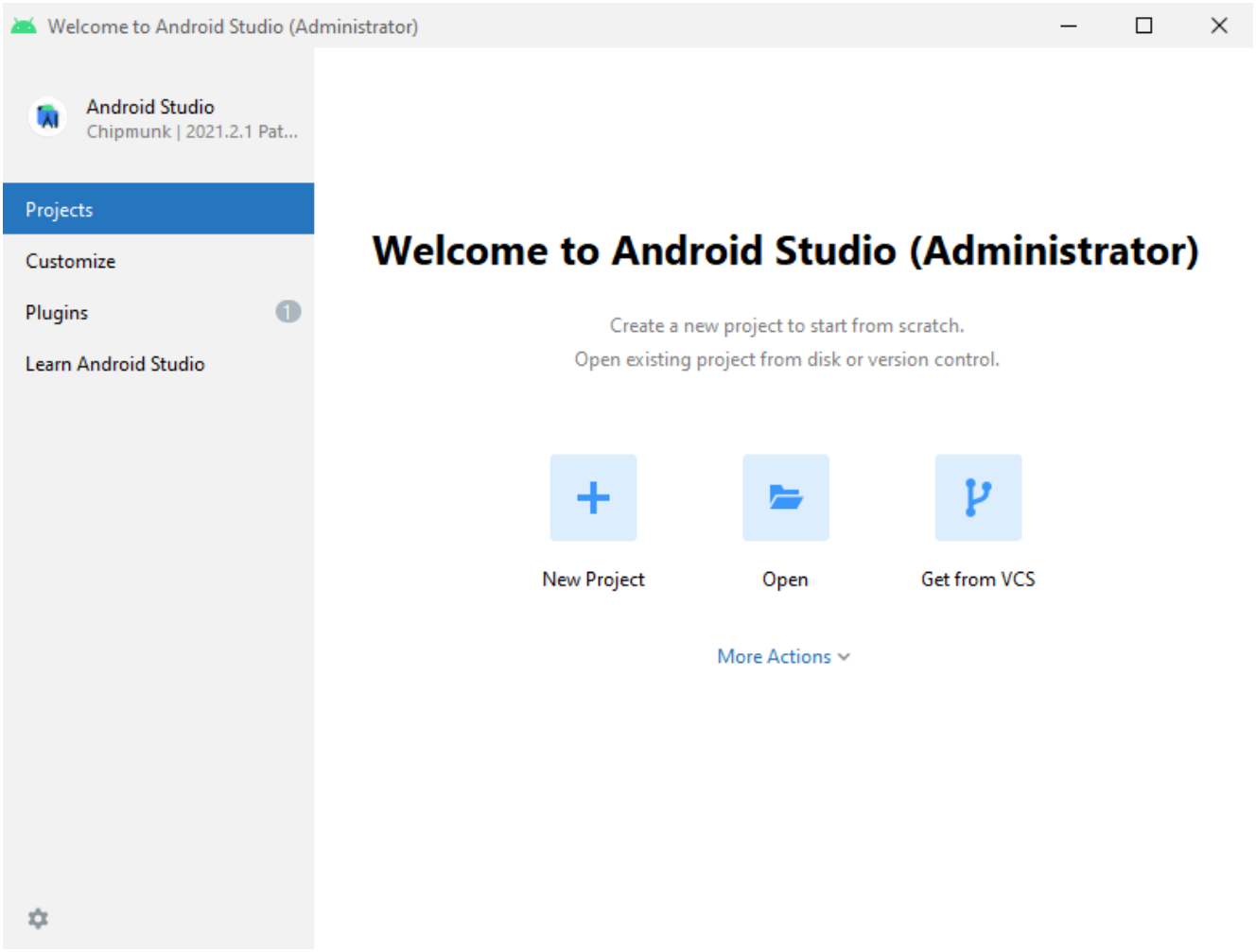

Android Studio Setup Wizard
Select UI Theme
Darcula
Light
module
src
HelloWorld
HelloWorld.java
import javax.swing.*;
import javax.awt.*;
public class HelloWorld {
public HelloWorld() {
JFrame frame = new JFrame ("Hello wo
JLabel label = new JLabel();
label.setFont(new Font("Serif", Font
Breakpoints
Line Breakpoints
Previous
Next
Cancel
Finish


Android Studio Setup Wizard
Verify Settings
If you want to review or change any of your installation settings, click Previous.
Current Settings:
Standard
SDK Folder:
C:\Users\LENOVO\AppData\Local\Android\Sdk
JDK Location:
C:\Program Files\Android\Android Studio\jre
Total Download Size:
455 MB
SDK Components to Download:
Android Emulator 329 MB
Android SDK Build-Tools 33 53 MB
Android SDK Platform 33 64.2 MB
Android SDK Platform-Tools 6.01 MB
Previous
Next
Cancel
Finish


Android Studio Setup Wizard
License Agreement
Read and agree to the licenses for the components which will be installed
Licenses
android-sdk-license
Android SDK Build-Tools 33
Android SDK Platform 33
SDK Patch Applier v4
Android Emulator
Android SDK Platform-Tools
intel-android-extra-license
Intel x86 Emulator Accelerator
Intel (R) Hardware Accelerated Execution Manager End-User License Agreement
Copyright (c) 2012 Intel Corporation. All rights reserved.
Redistribution. Redistribution and use in binary form, without modification, are permitted provided that the following conditions are met:
1.Redistributions must reproduce the above copyright notice and the following disclaimer in the documentation and/or other materials provided with the distribution.
2.Neither the name of Intel Corporation nor the names of its suppliers may be used to endorse or promote products derived from this software without specific prior written permission.
3.No reverse engineering, de-compilation, or disassembly of this software is permitted. Limited patent license. Intel Corporation grants a world-wide, royalty-free, non-exclusive license under patents it now or hereafter owns or controls to make, have made, use, import, offer to sell and sell ("Utilize") this software, but solely to the extent that any such patent is necessary to Utilize the software alone. The patent license shall not apply to any combinations which include this software. No hardware per se is licensed hereunder.
Decline
Accept
Previous
Next
Cancel
Finish


Android Studio Setup Wizard
Downloading Components
Starting download...
https://dl.google.com/android/repository/platform-33_r01.zip
7.6.5)".
Downloading https://dl.google
.com/android/repository/extras/intel/haxm-windows_v7_6_5.zip
"Install Intel x86 Emulator Accelerator (HAXM installer) (revision: 7.6.5)" ready.
Installing Intel x86 Emulator Accelerator (HAXM installer) in
C:\Users\LENOVO\AppData\Local\Android\Sdk\extras\intel
\Hardware_Accelerated_Execution_Manager
"Install Intel x86 Emulator Accelerator (HAXM installer) (revision: 7.6.5)" complete.
"Install Intel x86 Emulator Accelerator (HAXM installer) (revision: 7.6.5)" finished.
Preparing "Install Android SDK Platform 33 (revision: 1)".
Downloading https://dl.google.com/android/repository/platform-33_r01.zip
Previous
Next
Cancel
Finish


Android Studio Setup Wizard
Downloading Components
C:\Users\LENOVO\AppData\Local\Android\Sdk\platform-tools
"Install Android SDK Platform-Tools (revision: 33.0.2)" complete.
"Install Android SDK Platform-Tools (revision: 33.0.2)" finished.
Parsing C:\Users\LENOVO\AppData\Local\Android\Sdk\build-tools\33.0.0\package.xml
Parsing C:\Users\LENOVO\AppData\Local\Android\Sdk\emulator\package.xml
Parsing C:\Users\LENOVO\AppData\Local\Android\Sdk\extras\intel
\Hardware_Accelerated_Execution_Manager\package.xml
Parsing C:\Users\LENOVO\AppData\Local\Android\Sdk\patcher\v4\package.xml
Parsing C:\Users\LENOVO\AppData\Local\Android\Sdk\platform-tools\package.xml
Parsing C:\Users\LENOVO\AppData\Local\Android\Sdk\platforms\android-33\package.xml
Android SDK is up to date.
Running Intel® HAXM installer
Intel HAXM installed successfully!
Previous
Next
Cancel
Finish


Welcome to Android Studio (Administrator)
Android Studio
Chipmunk | 2021.2.1 Pat...
Projects
Customize
Plugins
Learn Android Studio
Welcome to Android Studio (Administrator)
Create a new project to start from scratch.
Open existing project from disk or version control.
New Project
Open
Get from VCS
More Actions

APPLICATION DESIGN

Model-View-Controller (MVC) is a graphical user interface (GUI) application paradigm that has recently taken hold. The concept involves the developer considering three main areas when developing an application. These areas can be kept as independent as possible. The model, loosely analogous to the process in the traditional Input-Process-Output (IPO) model, represents what the application does and the coding behind what it is intended to do.

When developing Android applications, we can easily isolate the view from the model and controller components. The view or layout of components is written in XML format in the main.xml file. Once the programmer determines which controls are necessary for the application, such as lists, text fields, buttons, and so on, he can plan and code their arrangement, size, labels, fonts, and colors in the main.xml file. Indeed, after the application is written, should the programmer decide he does not like the user interface's aesthetics, he can make changes without altering the Java classes and methods that produce the model and controller components of the MVC model.

Following are the rules for XML files:

1. All XML elements must have opening and closing tags.
2. XML tags are case sensitive.
3. XML elements must be properly nested.

ex.

```
<tag1>
        <tag2>
        </tag2>
</tag1>
```

4. XML documents must have a root element. A single tag pair must surround all other elements of the document. This is the root element.
5. XML attribute values must be quoted using either single or double quotes.

The Screen Layout and the main.xml File

As you have seen already, Eclipse creates a functional application as soon as you create a new project; you don't have to do anything. You have also seen that the screen configuration is controlled by the main.xml file that Eclipse generates. Take a moment to study the Eclipse-generated main.xml file here:

Android Layout Types

There are number of Layouts provided by Android which you will use in almost all the Android applications to provide different view, look and feel.

S.No	Layout & Description
1	**Linear Layout** LinearLayout is a view group that aligns all children in a single direction, vertically or horizontally.
2	**Relative Layout** RelativeLayout is a view group that displays child views in relative positions.
3	**Table Layout** TableLayout is a view that groups views into rows and columns.
4	**Absolute Layout** AbsoluteLayout enables you to specify the exact location of its children.
5	**Frame Layout** The FrameLayout is a placeholder on screen that you can use to display a single view.
6	**List View** ListView is a view group that displays a list of scrollable items.
7	**Grid View** GridView is a ViewGroup that displays items in a two-dimensional, scrollable grid.

Layout Attributes

Each layout has a set of attributes which define the visual properties of that layout. There are few common attributes among all the layouts and there are other attributes which are specific to that layout. Following are common attributes and will be applied to all the layouts:

Sr.No	Attribute & Description
1	**android:id** This is the ID which uniquely identifies the view.
2	**android:layout_width** This is the width of the layout.

3	**android:layout_height** This is the height of the layout
4	**android:layout_marginTop** This is the extra space on the top side of the layout.
5	**android:layout_marginBottom** This is the extra space on the bottom side of the layout.
6	**android:layout_marginLeft** This is the extra space on the left side of the layout.
7	**android:layout_marginRight** This is the extra space on the right side of the layout.
8	**android:layout_gravity** This specifies how child Views are positioned.
9	**android:layout_weight** This specifies how much of the extra space in the layout should be allocated to the View.
10	**android:layout_x** This specifies the x-coordinate of the layout.
11	**android:layout_y** This specifies the y-coordinate of the layout.
12	**android:layout_width** This is the width of the layout.
13	**android:layout_width** This is the width of the layout.
14	**android:paddingLeft** This is the left padding filled for the layout.

15	**android:paddingRight** This is the right padding filled for the layout.
16	**android:paddingTop** This is the top padding filled for the layout.
17	**android:paddingBottom** This is the bottom padding filled for the layout.

Component IDs

A view object may have a unique ID assigned to it which will identify the View uniquely within the tree. The syntax for an ID, inside an XML tag is −

android:id="@+id/my_button"

Following is a brief description of @ and + signs −

- The at-symbol (@) at the beginning of the string indicates that the XML parser should parse and expand the rest of the ID string and identify it as an ID resource.
- The plus-symbol (+) means that this is a new resource name that must be created and added to our resources. To create an instance of the view object and capture it from the layout, use the following −

Button myButton = (Button) findViewById(R.id.my_button);

A Few Simple Controls

activity_main.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="match_parent"
   android:orientation="vertical"
   android:gravity="center"
   tools:context=".MainActivity">
```

```
<Button
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="Display Your Name"
   android:id="@+id/button"/>
</LinearLayout>
```

MainActivity.java

```
package com.example.toastmessage;
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 button;
   @Override
   protected void onCreate(Bundle savedInstanceState)
{
      super.onCreate(savedInstanceState);
      setContentView(R.layout.activity_main);
      button=findViewById(R.id.button);
      button.setOnClickListener(new View.OnClickListener()
{
         @Override
         public void onClick(View v)
{
    Toast.makeText(MainActivity.this, "Mohamed Zamam Nazar", Toast.LENGTH_LONG).show();
         }
      });    }
}
```

CHECKBOX

1. Start android studio application.
2. Select File in menu bar.
3. Choose your project as **Empty Activity** and click Next button.
4. Name your project (The application name begins with an uppercase letter), select language as java, select your minimum API level as (Android 4.0 IceCreamSandwich) then click Finish button.
5. For designing, type xml coding in **activity_main.xml**.

   Components used for designing

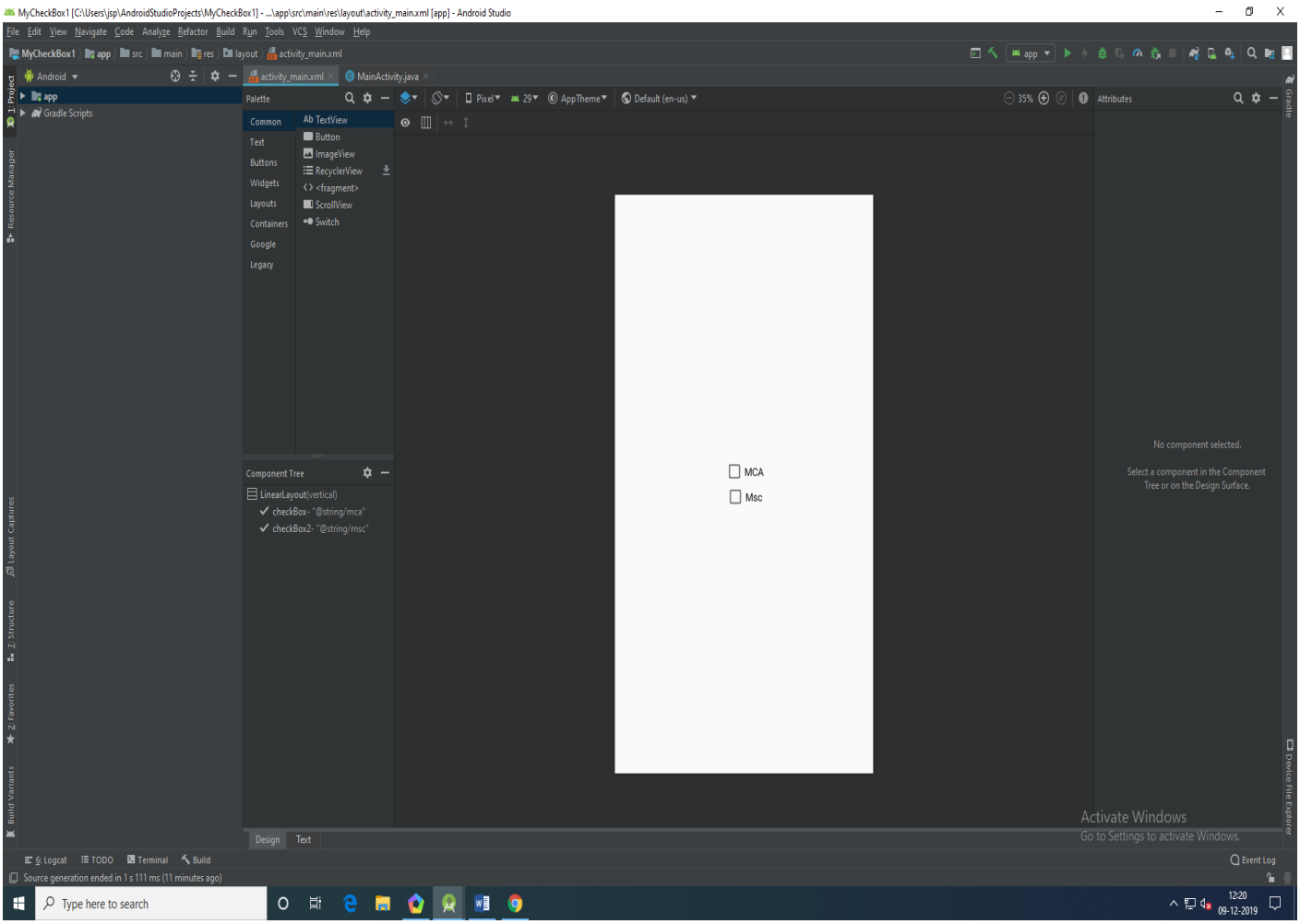
   - CheckBox
6. Type java coding in **MainActivity.java**.
7. To run the 'app' press play button in android studio or shift + F10.
8. Select Deployment project as your Emulator
9. In connected device, select your Emulator and click ok button.
10. Open your installed app in Emulator.

PROGRAM

activity_main.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="match_parent"
   android:orientation="vertical"
   android:gravity="center"
   android:padding="15dp"
   tools:context=".MainActivity">

   <CheckBox
      android:id="@+id/checkBox"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="@string/mca" />


   <CheckBox
      android:id="@+id/checkBox2"
      android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:text="@string/msc" />
</LinearLayout>
```

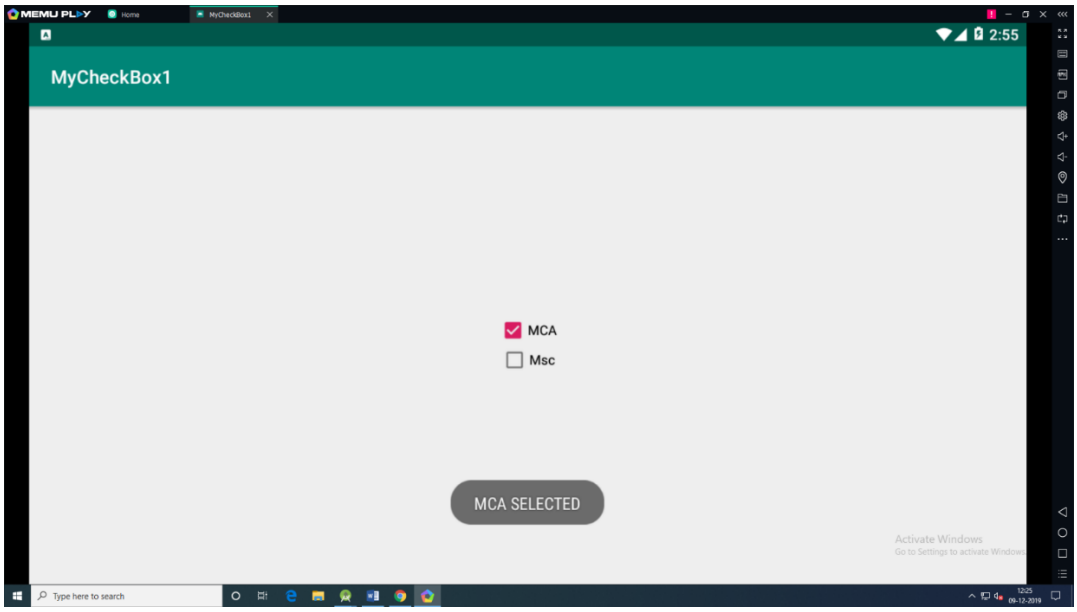


MainActivity.java

```
package com.example.mycheckbox1;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.CheckBox;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity

{
    CheckBox checkBox,checkBox2;

    @Override
    protected void onCreate(Bundle savedInstanceState)

{
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_main);
        checkBox=findViewById(R.id.checkBox);
        checkBox2=findViewById(R.id.checkBox2);
        checkBox.setOnClickListener(new View.OnClickListener()
{
            @Override
            public void onClick(View v) {
                if(checkBox.isChecked())
                {
Toast.makeText(getApplicationContext(), "MCA SELECTED", Toast.LENGTH_SHORT).show();
                }
                else
                {
Toast.makeText(getApplicationContext(),"MCA not SELECTED",Toast.LENGTH_SHORT).show();
                }
            }
        });
        checkBox2.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (checkBox2.isChecked())
                {
 Toast.makeText(getApplicationContext(), "MSC SELECTED", Toast.LENGTH_SHORT).show();
                }
                else
                {
  Toast.makeText(getApplicationContext(), "MSc Not Selected", Toast.LENGTH_SHORT).show();
                }
            }       });
    }}
```

OUTPUT



RADIOBUTTON

1. Start android studio application.
2. Select File in menu bar.
3. Choose your project as **Empty Activity** and click Next button.
4. Name your project (The application name begins with an uppercase letter), select language as java, select your minimum API level as (Android 4.0 IceCreamSandwich) then click Finish button.
5. For designing, type xml coding in **activity_main.xml**.

   Components used for designing

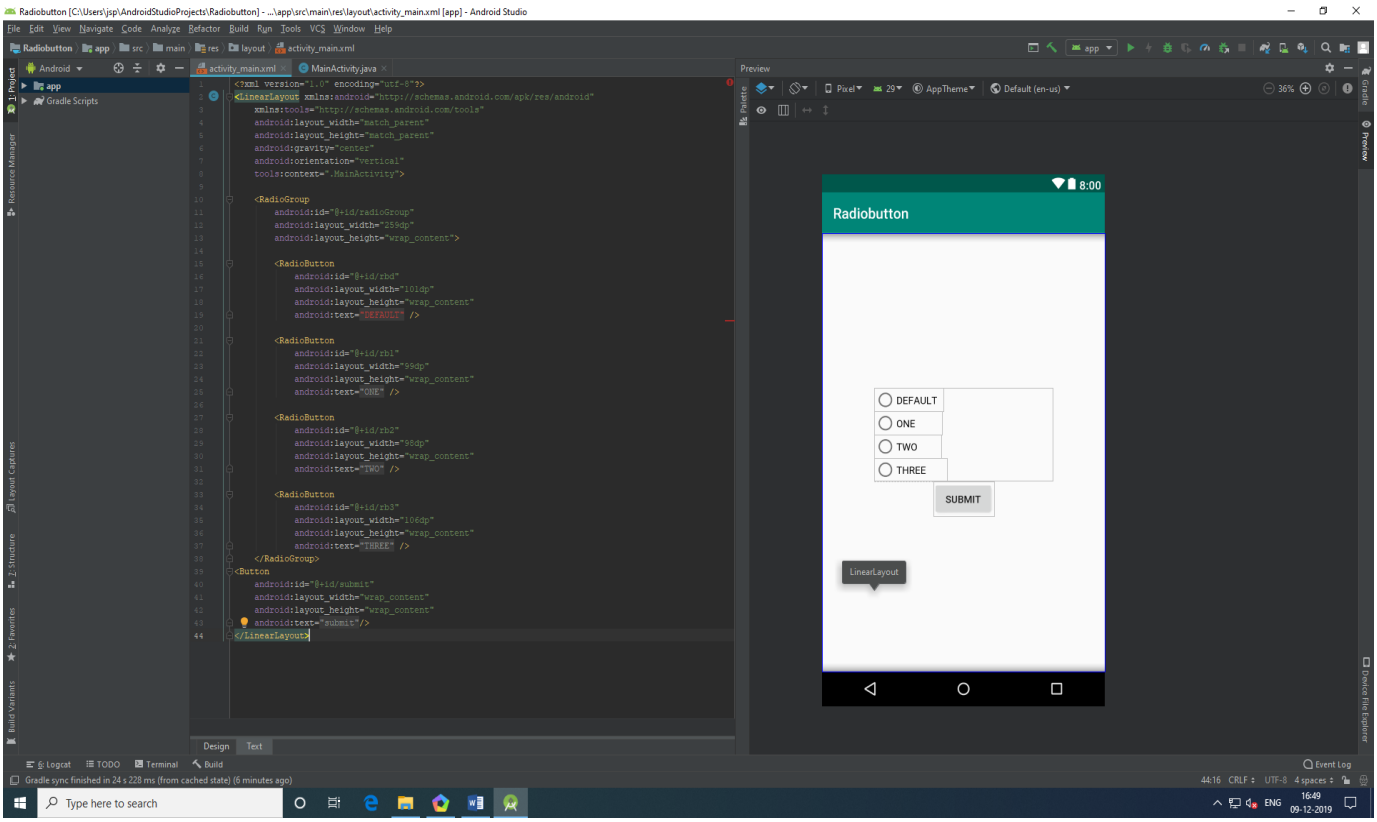
   - RadioGroup
   - RadioButton
   - Button
6. Type java coding in **MainActivity.java**.
7. To run the 'app' press play button in android studio or shift + F10.
8. Select Deployment project as your Emulator
9. In connected device, select your Emulator and click ok button.
10. Open your installed app in Emulator.

PROGRAM

activity_main.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="match_parent"
    android:gravity="center"
    android:orientation="vertical"
    tools:context=".MainActivity">

    <RadioGroup
        android:id="@+id/radioGroup"
        android:layout_width="259dp"
        android:layout_height="wrap_content">

        <RadioButton
            android:id="@+id/rbd"
            android:layout_width="101dp"
            android:layout_height="wrap_content"
            android:text="@string/default" />
```

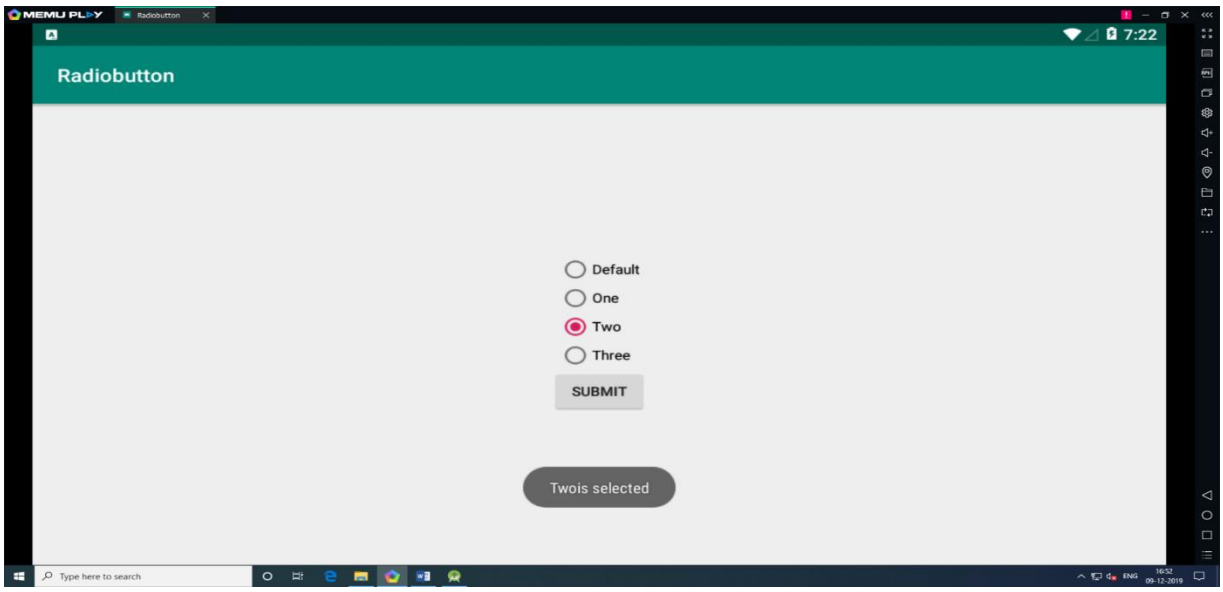
```
        <RadioButton
            android:id="@+id/rb1"
            android:layout_width="99dp"
            android:layout_height="wrap_content"
            android:text="@string/one" />

        <RadioButton
            android:id="@+id/rb2"
            android:layout_width="98dp"
            android:layout_height="wrap_content"
            android:text="@string/two" />

        <RadioButton
            android:id="@+id/rb3"
            android:layout_width="106dp"
            android:layout_height="wrap_content"
            android:text="@string/three" />
    </RadioGroup>
<Button
    android:id="@+id/submit"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/submit"/>
</LinearLayout>
```



MainActivity.java

```
package com.example.radiobutton;
import androidx.appcompat.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.Toast;
public class MainActivity extends AppCompatActivity implements View.OnClickListener {
    RadioGroup radioGroup;
    RadioButton radioButton;
    Button submit;
    @Override
    protected void onCreate(Bundle savedInstanceState)

 {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        submit = findViewById(R.id.submit);
        submit.setOnClickListener(this);
    }
    @Override
    public void onClick(View v)

{
        radioGroup=findViewById(R.id.radioGroup);
        int id =radioGroup.getCheckedRadioButtonId();
        radioButton=findViewById(id);
        Toast.makeText(this, radioButton.getText()+"is selected", Toast.LENGTH_SHORT).show();
    }
}
```

OUTPUT



SPINNER

PROCEDURE

1. Start android studio application.
2. Select File in menu bar.
3. Choose your project as **Empty Activity** and click Next button.
4. Name your project (The application name begins with an uppercase letter), select language as java, select your minimum API level as (Android 4.0 IceCreamSandwich) then click Finish button.
5. For designing, type xml coding in **activity_main.xml**.

   Components used for designing

   - Spinner
6. Type the string.xml coding in this (Project - app – res – values – string.xml) location.
7. Type java coding in **MainActivity.java**.
8. To run the 'app' press play button in android studio or shift + F10.
9. Select Deployment project as your Emulator
10. In connected device, select your Emulator and click ok button.
11. Open your installed app in Emulator.

PROGRAM

Activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout
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="com.example.application.spinnerexample.MainActivity">

   <Spinner
      android:id="@+id/spinner1"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"/>

</android.support.constraint.ConstraintLayout>
```

String.xml

```
<resources>
   <string name="app_name">Spinner Example</string>
```

```
    <string-array name="numbers">
        <item>One</item>
        <item>Two</item>
        <item>Three</item>
        <item>Four</item>
        <item>Five</item>
    </string-array>
</resources>
```

Mainactivity.java

```
package com.example.application.spinnerexample;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.Spinner;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity implements
AdapterView.OnItemSelectedListener {

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

        Spinner spinner = findViewById(R.id.spinner1);
        ArrayAdapter<CharSequence> adapter = ArrayAdapter.createFromResource(this,
                R.array.numbers, android.R.layout.simple_spinner_item);
        adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        spinner.setAdapter(adapter);
        spinner.setOnItemSelectedListener(this);
    }

    @Override
    public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
        String text = parent.getItemAtPosition(position).toString();
        Toast.makeText(parent.getContext(), text, Toast.LENGTH_SHORT).show();
    }

    @Override
    public void onNothingSelected(AdapterView<?> parent) {
    }
}
```

DATE PICKER

AIM

To develop an android program using the Date Picker.

PROCEDURE

1. Start android studio application.
2. Select File in menu bar.
3. Choose your project as **Empty Activity** and click Next button.
4. Name your project (The application name begins with an uppercase letter), select language as java, select your minimum API level as (Android 4.0 IceCreamSandwich) then click Finish button.
5. For designing, type xml coding in **activity_main.xml**.

   Components used for designing

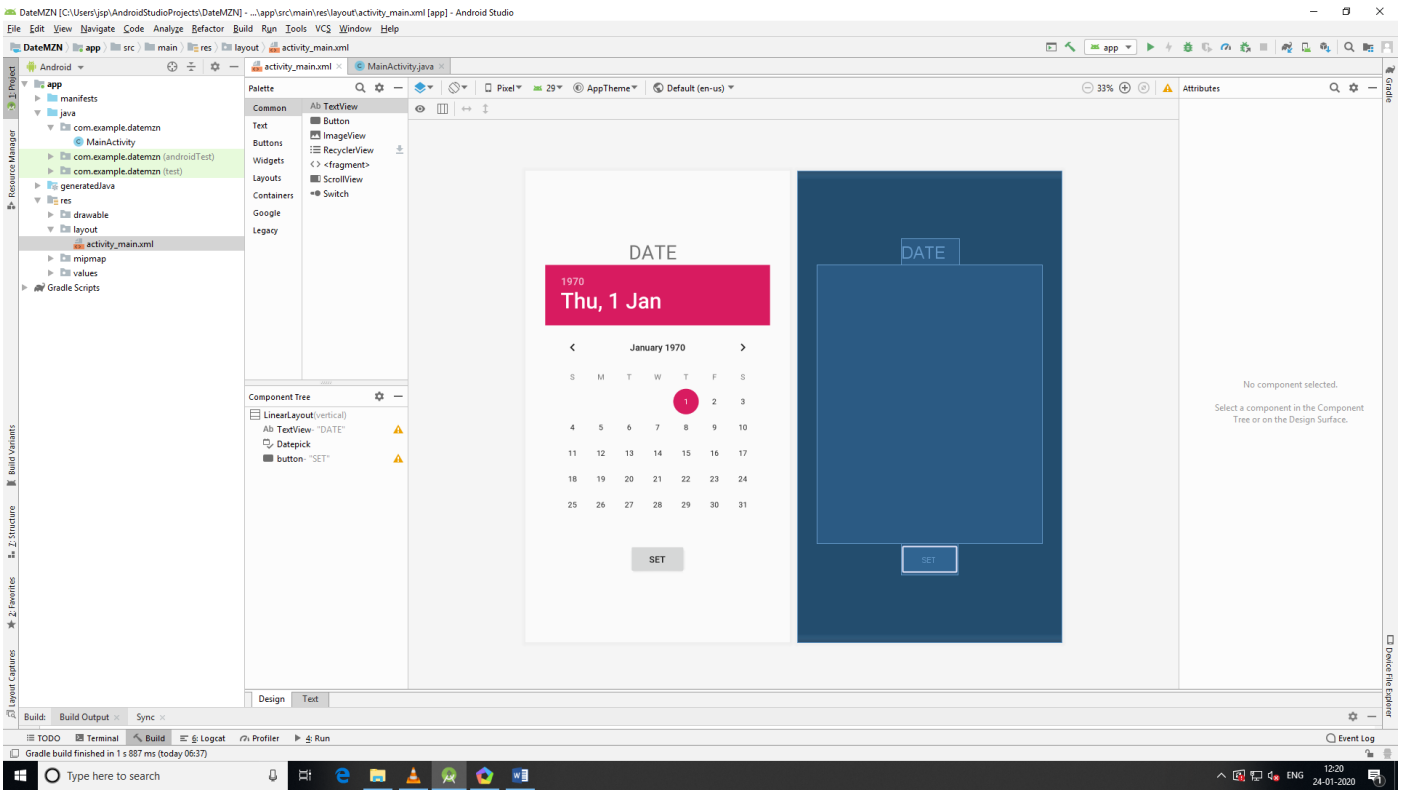
   - TextView
   - DatePicker
   - Button
6. Type the string.xml coding in this (Project - app – res – values – string.xml) location.
7. Type java coding in **MainActivity.java**.
8. To run the 'app' press play button in android studio or shift + F10.
9. Select Deployment project as your Emulator
10. In connected device, select your Emulator and click ok button.
11. Open your installed app in Emulator.

PROGRAM

Activity_main.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="match_parent"
   android:orientation="vertical"
   android:gravity="center"
   tools:context=".MainActivity">
```

```
<TextView
        android:layout_width="90dp"
        android:layout_height="wrap_content"
        android:text="DATE"
        android:textSize="30sp"></TextView>

    <DatePicker
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/Datepick">
    </DatePicker>
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/button"
        android:text="SET"/>

</LinearLayout>
```

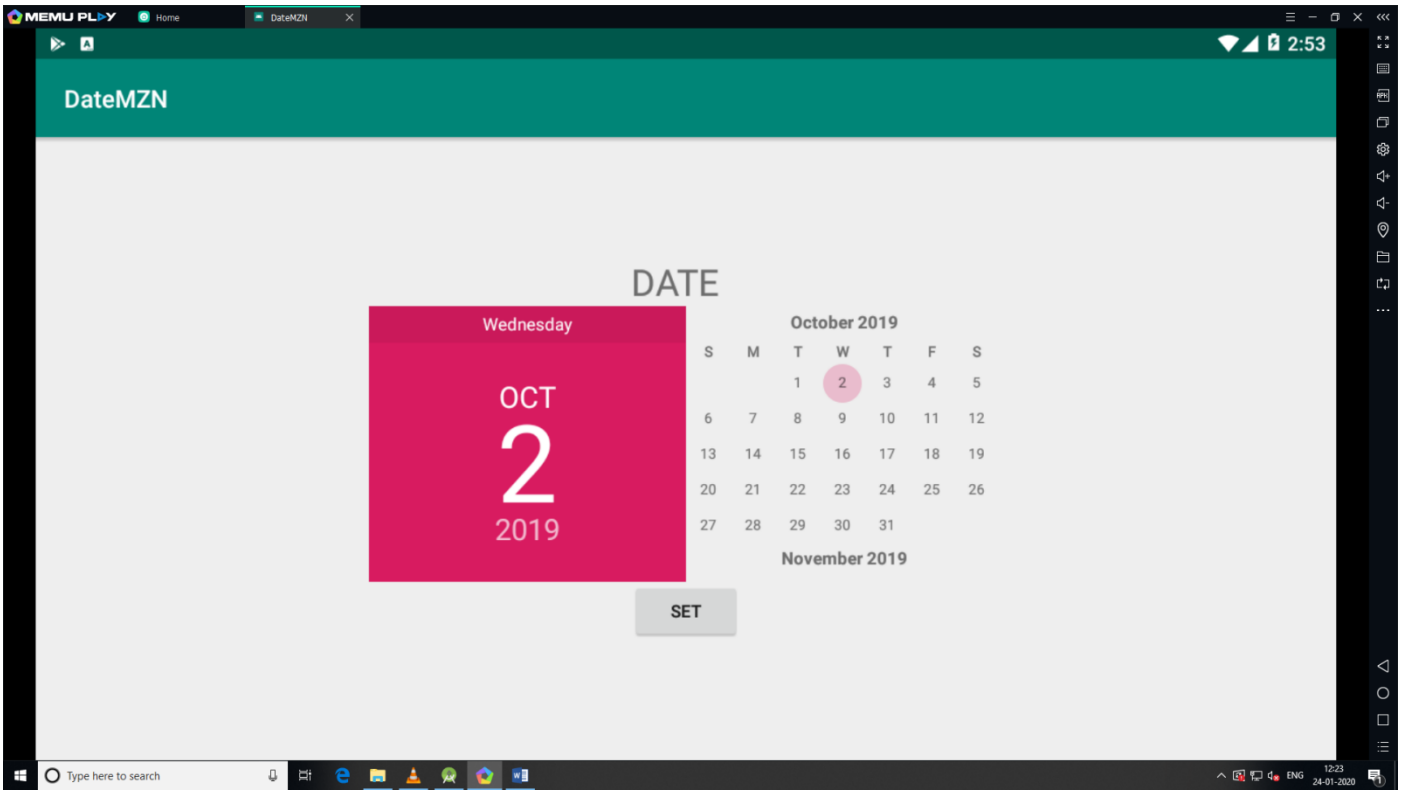


MainActivity.java

```
package com.example.datemzn;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
```

```
import android.widget.DatePicker;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {
    DatePicker Datepick;
    Button button;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Datepick =findViewById(R.id.Datepick);
        button =findViewById(R.id.button);
        button.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Toast.makeText(MainActivity.this,
Datepick.getDayOfMonth()+"/"+Datepick.getMonth()+"/"+Datepick.getYear(),
Toast.LENGTH_SHORT).show();
            }
        });
    }
}
```

OUTPUT



BASIC GRAPHICS BY EXTENDING THE VIEW CLASS

Combining Graphics with a Touch Listener

Up to this point we have been using widgets included in the Android software development kit (SDK). Sometimes you will want to build custom widgets that will actually extend a built-in base class. After all, this is one of the advantages of using an object-oriented programming language. The following two examples will extend the View class to become something uniquely suited to our purpose.

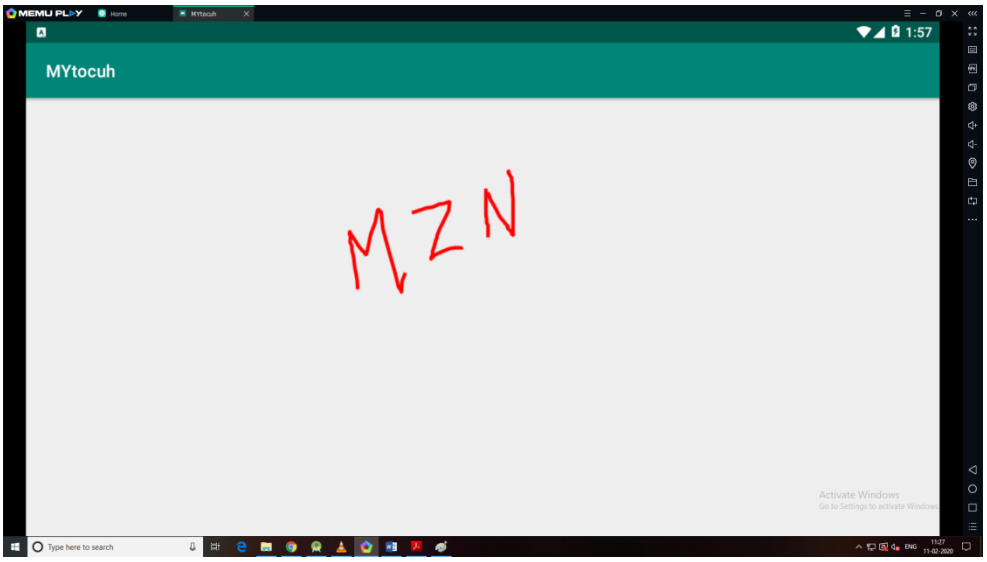
MainActivity.java

```
package com.example.mytocuh;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;

public class MainActivity extends AppCompatActivity {
MyCanvas myCanvas;

   @Override
   protected void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      myCanvas=new MyCanvas(this,null);
      setContentView(myCanvas);
   }
}
```

MyCanvas.java

```
package com.example.mytocuh;

import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Color;
import android.graphics.Paint;
import android.graphics.Path;
import android.util.AttributeSet;
import android.view.MotionEvent;
import android.view.View;

public class MyCanvas extends View
{
   Paint paint;
   Path path;

   public MyCanvas(Context context, AttributeSet attrs) {
      super(context, attrs);
      paint=new Paint();
      path=new Path();
      paint.setAntiAlias(true);
      paint.setColor(Color.RED);
```

```
        paint.setStrokeJoin(Paint.Join.ROUND);
        paint.setStyle(Paint.Style.STROKE);
        paint.setStrokeWidth(5f);
    }

    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);
        canvas.drawPath(path,paint);
    }

    @Override
    public boolean onTouchEvent(MotionEvent event) {
        float xpos=event.getX();
        float ypos=event.getY();
        switch(event.getAction())
        {
            case MotionEvent.ACTION_DOWN:
                path.moveTo(xpos,ypos);
                return true;
            case MotionEvent.ACTION_MOVE:
                path.lineTo(xpos,ypos);
                break;
            case MotionEvent.ACTION_UP:
                break;
                default:
                    return false;
        }
        invalidate();
        return true;
    }
}
```



Canvas

Class	`Canvas`
Package	`android.graphics`
Extends	`java.lang.Object`
Overview	Holds the "draw" calls that write to a `Bitmap`

Methods used in this chapter:

`void drawCircle(float cx, float cy, float radius, Paint paint)`	Draws a circle at the specified x,y coordinates with the specified radius and other features specified by the `Paint` object variable, such as the color

Other commonly used methods:

`void drawLine(float startX, float startY, float stopX, float stopY, Paint paint)`	Draws a line from start x,y coordinates to stop x,y coordinates using features specified in the `Paint` object variable, such as the color
`void drawRect(float left, float top, float right, float bottom, Paint paint)`	Draws a rectangle from top, left to bottom, right using features specified in the `Paint` object variable, such as the color
`void drawText(String text, float x, float y, Paint paint)`	Draws text starting at the x,y coordinates using features specified in the Paint object variable such as the color
`void drawCircle(float cx, float cy, float radius, Paint paint)`	Draw the specified circle using the specified paint
`void drawColor(int color)`	File the entire canvas bitmap (restricted to the current clip) with the specified color
`void drawPicture(Picture picture)`	Save the canvas state, draw the picture, and then restore the canvas state

MotionEvent

Class	`MotionEvent`
Package	`android.view`
Extends	`Android.view.InputEvent`
Overview	Used to report movement

Methods used in this chapter:

n/a

Symbolic constants used:

`ACTION_DOWN`	Signals the start of a gesture. Contains the starting location.
`ACTION_MOVE`	Signals a change has happened since `ACTION_DOWN`.
`ACTION_UP`	Signals a gesture has finished. Contains the release location.

Paint

Class	`Paint`
Package	`android.graphics`
Extends	`java.lang.Object`
Overview	Holds the "draw" calls that write to a `Bitmap`

Methods used in this chapter:

`setColor(int color)`	Sets the `Paint`'s color.
`setStyle(Paint.Style style)`	Sets the `Paint`'s style. `Paint.Style` sets how the line is filled, stroked, or both. Three choices are available: FILL, FILL_AND_STROKE, and STROKE.

Other commonly used methods:

`setTextSize(float size)`	Sets the `Paint`'s text size
`int getColor( )`	Returns the `Paint`'s current color setting
`float measureText(String text)`	Returns the width of the text
`void setStrokeWidth(float width)`	Sets the width for stroking

Bitmap

Class	`Bitmap`
Package	`android.graphics`
Extends	`java.lang.Object`
Overview	Holds the "draw" calls that write to a `Bitmap`

Methods used in this chapter:

`boolean compress(Bitmap.CommpressFormat format, int quality, OutputStream stream)`	Writes a compressed version of the `Bitmap` to the output stream

Other commonly used methods:

`void copyPixelsFromBuffer (Buffer src)`	Copy the pixels from the buffer, beginning at the current position, overwriting the `Bitmap`'s pixels
`static Bitmap createBitmap (Bitmap source)`	Returns an immutable `Bitmap` from the source `Bitmap`
`final int getByteCount()`	Returns the number of bytes used to store this `Bitmap`'s pixels
`int getDensity()`	Returns the density for this `Bitmap`
`final int getHeight()`	Returns the height of this `Bitmap`
`final int getWidth()`	Returns the width of this `Bitmap`
`void writeParcel(Parcel p, int flags)`	Writes this `Bitmap` and its pixels to the specified parcel

POPUP DIALOG BOXES

AIM

To develop an android program using the Dialog box.

PROCEDURE

1. Start android studio application.
2. Select File in menu bar.
3. Choose your project as **Empty Activity** and click Next button.
4. Name your project (The application name begins with an uppercase letter), select language as java, select your minimum API level as (Android 4.0 IceCreamSandwich) then click Finish button.
5. For designing, type xml coding in **activity_main.xml**.

   Components used for designing

   - TextView
   - Button
6. Type java coding in **MainActivity.java**.
7. To run the 'app' press play button in android studio or shift + F10.
8. Select Deployment project as your Emulator
9. In connected device, select your Emulator and click ok button.
10. Open your installed app in Emulator.

PROGRAM

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:orientation="vertical"
   android:gravity="center"
   tools:context=".MainActivity">

   <TextView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="DIALOG ALERT"/>
```

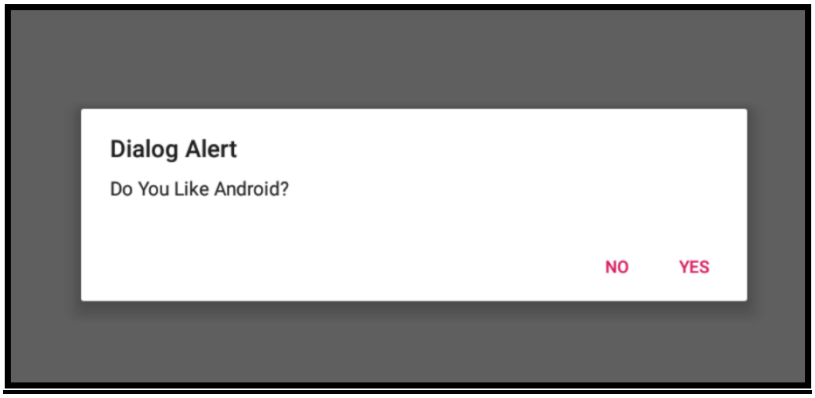
```
    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/button"
        android:text="Click On Me"
        android:onClick="showAlertDialog"/>

</LinearLayout>
```

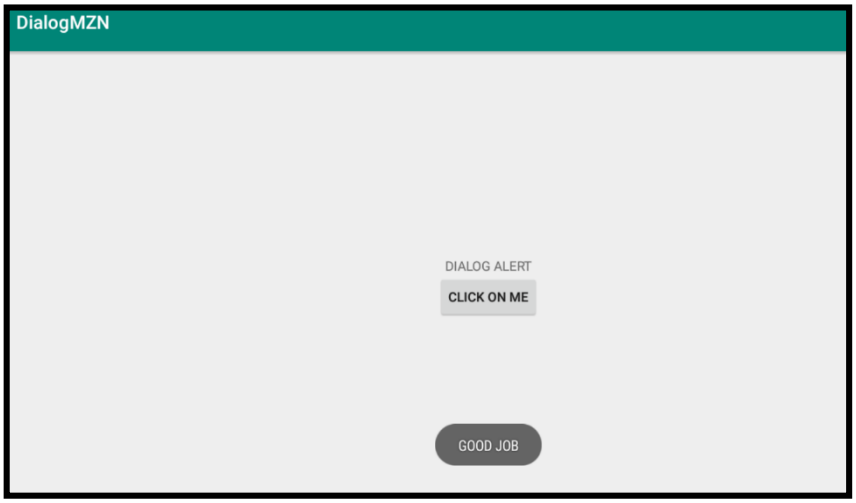
MainActivity.java

```
package com.example.dialogmzn;

import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;

import android.content.DialogInterface;
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 showAlertDialog(View V)
    {
```

```
        AlertDialog.Builder alert=new AlertDialog.Builder(this);
        alert.setTitle("Dialog Alert");
        alert.setMessage("Do You Like Android?");
        alert.setPositiveButton("YES", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                Toast.makeText(MainActivity.this, "GOOD JOB", Toast.LENGTH_SHORT).show();
            }
        });
        alert.setNegativeButton("NO", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                Toast.makeText(MainActivity.this, "TOO BAD", Toast.LENGTH_SHORT).show();
            }
        });
        alert.create().show();
    }
}
```

OUTPUT





AlertDialog

Class	`AlertDialog`
Package	`android.app`
Extends	`android.app.Dialog`
Overview	A subclass of `Dialog` that can contain one, two, or three buttons. A string can be displayed using the `setMessage( )` method.

Methods used in this chapter:

`void setTitle(CharSequence title)`	Sets the title for this `AlertDialog`

Other commonly used methods:

`void setMessage(CharSequence message)`	Sets the message for this `AlertDialog`
`void setIcon(Drawable icon)`	Sets the icon to appear on this `AlertDialog`
`void setView(View view)`	Sets the `View` (or child of the `View` class) to be displayed on this `AlertDialog`
`Button getButton(int whichButton)`	Gets one of the `Button` objects used in the dialog, using `whichButton` as an index value
`ListView getListView( )`	Gets the `ListView` object used in the dialog

AlertDialog.Builder

Class	`AlertDialog.Builder`
Package	`android.app`
Extends	`java.lang.Object`
Overview	Class for creating `AlertDialogs`

Methods used in this chapter:

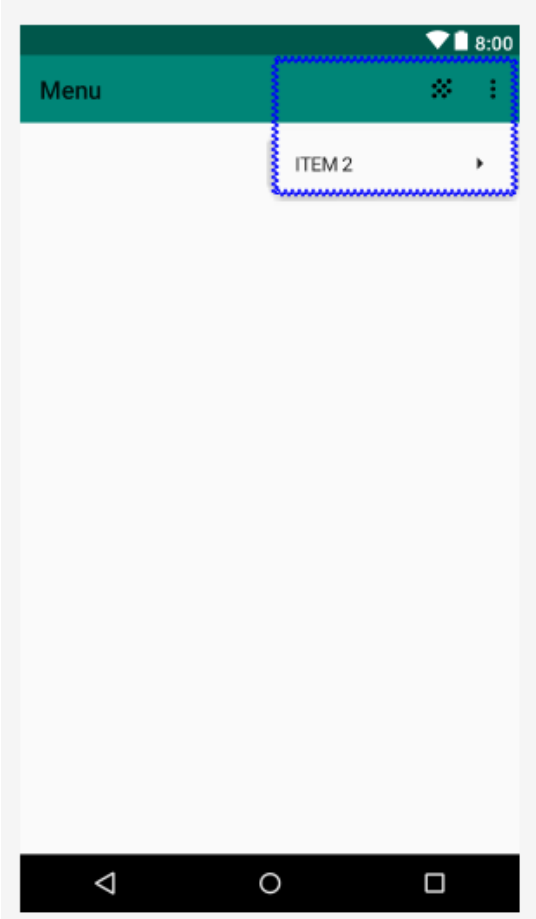
`AlertDialog create( )`	Creates an `AlertDialog` with the arguments supplied to this `Builder`

Other commonly used methods:

`void setTitle(CharSequence title)`	Sets the title for the `AlertDialog(s)` to be produced
`void setMessage(CharSequence message)`	Sets message for the `AlertDialog(s)` to be produced
`void setIcon(Drawable icon)`	Sets the icon for the `AlertDialog(s)` to be produced
`void setView(View view)`	Sets the `View` (or child of the `View` class) for the `AlertDialog(s)` to be produced
`void show( )`	Builds, then shows an `AlertDialog`

MENUS ON THE ANDROID DEVICES
example_menu.xml

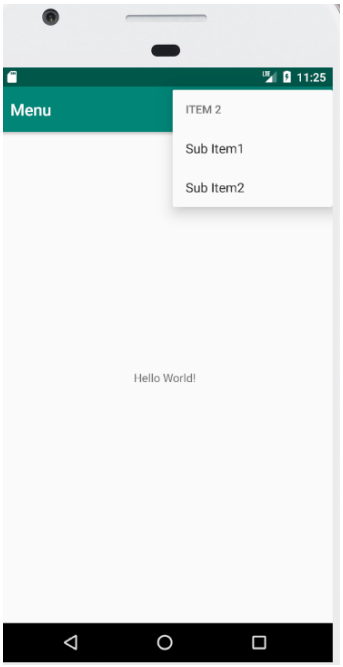
```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto">
   <item android:id="@+id/item1"
      android:icon="@drawable/ic_icon"
      android:title="ITEM 1"
      app:showAsAction="ifRoom" />
   <item android:id="@+id/item2"
      android:title="ITEM 2"
      app:showAsAction="never">
      <menu>
         <item android:title="Sub Item1"
            android:id="@+id/subitem1"/>
         <item android:title="Sub Item2"
            android:id="@+id/subitem2"/>
      </menu>
   </item>
</menu>
```



MainActivity.java

```
package com.example.menu;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity {

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

   @Override
   public boolean onCreateOptionsMenu(Menu menu) {
      MenuInflater inflater=getMenuInflater();
      inflater.inflate(R.menu.example_menu,menu);
      return true;
   }

   @Override
   public boolean onOptionsItemSelected(@NonNull MenuItem item) {
      switch(item.getItemId())
      {
         case R.id.item1:
            Toast.makeText(this, "ITEM1 SELECTED", Toast.LENGTH_SHORT).show();
            return true;
         case R.id.item2:
            Toast.makeText(this, "ITEM2 SELECTED", Toast.LENGTH_SHORT).show();
            return true;
         case R.id.subitem1:
            Toast.makeText(this, "SUB ITEM1 SELECTED", Toast.LENGTH_SHORT).show();
            return true;
         case R.id.subitem2:
            Toast.makeText(this, "SUB ITEM2 SELECTED", Toast.LENGTH_SHORT).show();
            return true;
            default:
               return super.onOptionsItemSelected(item);
      }
   }
}
```

OUTPUT



Menu

Class	Menu (interface)
Package	android.menu
Extends	n/a
Overview	An interface for managing items on a menu. Note: If six or more menu choices are necessary, they can be reached by the More item on the icon menu. However, they will not show icons, and item check marks are discouraged.

Methods used in this chapter:

n/a

Other commonly used methods:

`abstract MenuItem add (CharSequence title)`	Adds a new menu item to the menu
`abstract SubMenu addSubMenu (int groupid, int itemid, int order, CharSequence title)`	Adds a new submenu to the menu
`abstract void close( )`	Closes the menu
`abstract void clear( )`	Removes all existing items from the menu, leaving it empty as if it had just been created
`abstract MenuItem getItem(int index)`	Gets the menu item at the given index.
`abstract MenuItem findItem(int id)`	Gets the menu item with a particular identifier
`abstract int size( )`	Returns the number of items in the menu

MenuInflater

Class	`MenuInflater`
Package	`android.menu`
Extends	`java.lang.Object`
Overview	This class is used to convert XML files into menus.

Methods used in this chapter:

`void inflate(int menuRes, Menu menu)`	Inflate a menu hierarchy from the specified XML resource. The `menuRes` value refers to the integer value found in the `R.java` file associated with the `menu.xml file`.

MenuItem

Class	`MenuItem` (interface)
Package	`android.view`
Extends	n/a
Overview	Interface for direct access to a previously created menu item

Methods used in this chapter:

`abstract int getItemId( )`	Returns the identifier for this menu item.

Other commonly used methods:

`abstract intent getIntent( )`	Returns the `Intent` associated with this menu item
`abstract int getOrder( )`	Returns the category and order within the category for this menu item
`abstract CharSequence getTitle( )`	Returns the current title of this menu item
`abstract SubMenu getSubMenu( )`	Returns the `SubMenu` for this menu item, if it has one
`abstract MenuItem setIcon (Drawable icon)`	Change the icon associated with this item
`abstract MenuItem setIntent (Intent intent)`	Change the `Intent` associated with this item
`abstract MenuItem setNumeric Shortcut(char numericChar)`	Change the numeric shortcut associated with this item

WORKING WITH IMAGES

Displaying Images

- Displaying images on the Android devices is fairly simple.
- Android supports the four common image formats
  - ✓ PNG
  - ✓ JPG
  - ✓ BMP
  - ✓ GIF
- Images that you want with your application are stored in the **res** directory.
- You can create a sub-directory under the **res** directory called **drawable.**
- Note that all the image file names should be in lowercase and that contains only letters, numbers and underscores.
- The device will adjust the size of the image and maintain the aspect ratio when the image is displayed.
- Image can be assigned to an ImageView either in the xml file or in the java code.
- After the images in this folder, a resource ID will be generated automatically.

Change images randomly using buttons

Arrays.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
   <array name="icons">
      <item>@drawable/img1</item>
      <item>@drawable/img2</item>
      <item>@drawable/img3</item>
   </array>
   </resources>
```

Activity_main.xml

```
<?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">

   <ImageView
      android:id="@+id/imgv1"
      android:layout_width="343dp"
      android:layout_height="318dp"
      android:layout_alignParentStart="true"
      android:layout_alignParentLeft="true"
      android:layout_alignParentEnd="true"
      android:layout_alignParentRight="true"
      android:layout_alignParentBottom="true"
      android:layout_marginStart="36dp"
```

```
        android:layout_marginLeft="36dp"
        android:layout_marginEnd="33dp"
        android:layout_marginRight="33dp"
        android:layout_marginBottom="340dp" />

    <Button
        android:id="@+id/b1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentStart="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentEnd="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentBottom="true"
        android:layout_marginStart="138dp"
        android:layout_marginLeft="138dp"
        android:layout_marginEnd="146dp"
        android:layout_marginRight="146dp"
        android:layout_marginBottom="175dp"
        android:text="CHANGE IMAGE" />


</RelativeLayout>
```

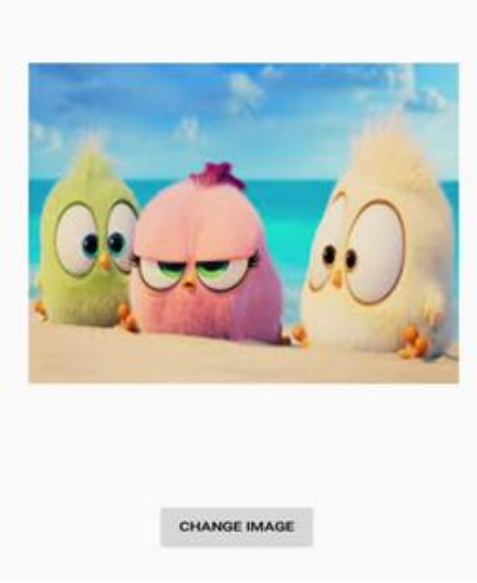
DESIGN

MainActivity.java

```
package com.example.images;

import androidx.appcompat.app.AppCompatActivity;

import android.content.res.TypedArray;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
```

```
import java.util.Random;

public class MainActivity extends AppCompatActivity {
        TypedArray icons;
        ImageView imgv1;
        Button b1;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        icons=getResources().obtainTypedArray(R.array.icons);
        imgv1=findViewById(R.id.imgv1);
        b1=findViewById(R.id.b1);
        b1.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                changeImage();
            }
        });
    }
    private void changeImage()
    {
        Random random=new Random();
        int number=random.nextInt(3);
        imgv1.setBackgroundResource(icons.getResourceId(number,1));
    }
}
```

OUTPUT



ZOOM THE IMAGE

PROCEDURE

1. Start android studio application.
2. Select File in menu bar.
3. Choose your project as **Empty Activity** and click Next button.
4. Name your project (The application name begins with an uppercase letter), select language as java, select your minimum API level as (Android 4.0 IceCreamSandwich) then click Finish button.
5. For designing, type xml coding in **activity_main.xml**.

   Components used for designing

   - ZoomControl
   - ImageView
6. Type java coding in **MainActivity.java**.
7. To run the 'app' press play button in android studio or shift + F10.
8. Select Deployment project as your Emulator
9. In connected device, select your Emulator and click ok button.
10. Open your installed app in Emulator.

PROGRAM

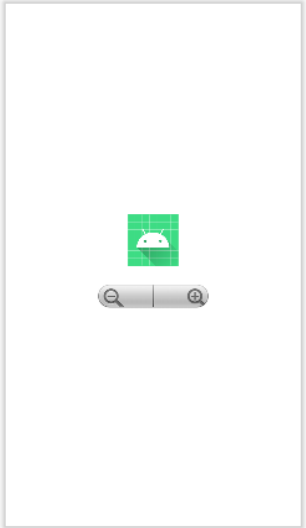
activity_main.xml

ATTRIBUTES	VALUES
LINEARLAYOUT	
Layout_width	Match parent
Layout_height	Match parent
Orientation	Vertical
Gravity	Center
IMAGEVIEW	
Layout_width	Wrap content
Layout_height	Wrap content
Id	Img
src	@mipmap/ic_launcher
ZOOMCONTROL	
Layout_width	Wrap content
Layout_height	Wrap content
Id	zio

```
package com.example.simple;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.ImageView;
import android.widget.ZoomControls;

public class MainActivity extends AppCompatActivity
{
   ZoomControls zio;
   ImageView img;

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

         img=findViewById(R.id.img);
         zio=findViewById(R.id.zio);
         zio.setOnZoomInClickListener(new View.OnClickListener()
         {
         @Override
         public void onClick(View view)
         {
            float x=img.getScaleX();
            float y=img.getScaleY();

            img.setScaleX(x+1);
            img.setScaleY(y+1);
         }
      });
      zio.setOnZoomOutClickListener(new View.OnClickListener()
         {
         @Override
         public void onClick(View view)
         {

            float x=img.getScaleX();
            float y=img.getScaleY();

            img.setScaleX(x-1);
            img.setScaleY(y-1);

         }
      });
```

```
        }
}
```

DESIGN

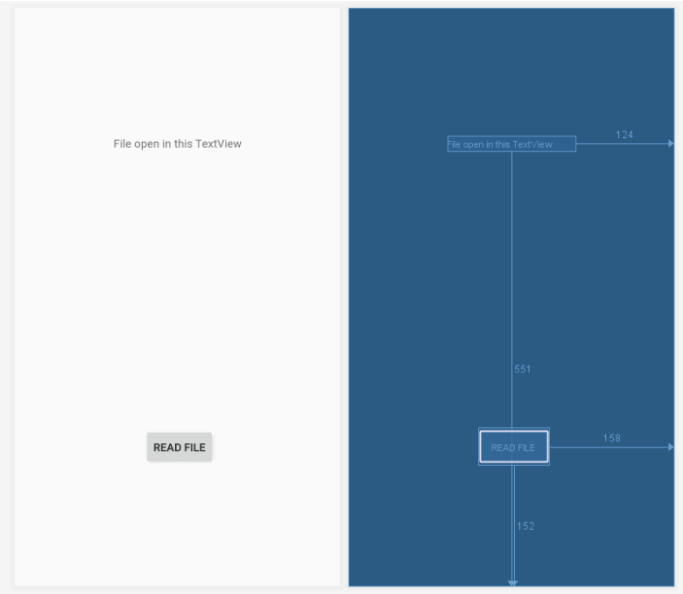


OUTPUT

WORKING WITH TEXT FILES

Activity_main.xml

```
<?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:id="@+id/tv1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentEnd="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentBottom="true"
        android:layout_marginEnd="124dp"
        android:layout_marginRight="124dp"
        android:layout_marginBottom="551dp"
        android:text="File open in this TextView" />

    <Button
        android:id="@+id/b1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentEnd="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentBottom="true"
        android:layout_marginEnd="158dp"
        android:layout_marginRight="158dp"
        android:layout_marginBottom="152dp"
        android:text="READ FILE" />
</RelativeLayout>
```



MainActivity.java

```
package com.example.textfile;

import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

import java.io.IOException;
import java.io.InputStream;

public class MainActivity extends AppCompatActivity {

    Button b1;
    TextView tv1;

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    tv1=findViewById(R.id.tv1);
    b1=findViewById(R.id.b1);
    b1.setOnClickListener(new View.OnClickListener() {
      @Override
      public void onClick(View v) {
        String text="";
        try {
          InputStream is=getAssets().open("text.txt");
          int size=is.available();
          byte[] buffer = new byte[size];
          is.read(buffer);
          is.close();;
          text=new String(buffer);
        }
        catch (IOException ex)
        {
          ex.printStackTrace();
        }
        tv1.setText(text);
      }
    });
  }
}
```

OUTPUT


12:11
71%
TextFile
<?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:id="@+id/tv1"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignParentEnd="true"
READ FILE

DATABASE USING SQLite

SQLite is an opensource SQL database that stores data to a text file on a device. Android comes in with built in SQLite database implementation.

SQLite supports all the relational database features. In order to access this database, you don't need to establish any kind of connections for it like JDBC, ODBC etc.

Database - Package

The main package is **android.database.sqlite** that contains the classes to manage your own databases

Database - Creation

In order to create a database you just need to call this method openOrCreateDatabase with your database name and mode as a parameter. It returns an instance of SQLite database which you have to receive in your own object. Its syntax is given below

```
SQLiteDatabase mydatabase = openOrCreateDatabase("your database
                                name",MODE_PRIVATE,null);
```

Apart from this, there are other functions available in the database package that does this job. They are listed below

Sr.No	Method & Description
1	openDatabase(String path, SQLiteDatabase.CursorFactory factory, int flags, DatabaseErrorHandler errorHandler) This method only opens the existing database with the appropriate flag mode. The common flags mode could be OPEN_READWRITE OPEN_READONLY
2	openDatabase(String path, SQLiteDatabase.CursorFactory factory, int flags) It is similar to the above method as it also opens the existing database but it does not define any handler to handle the errors of databases
3	openOrCreateDatabase(String path, SQLiteDatabase.CursorFactory factory) It not only opens but create the database if it not exists. This method is equivalent to openDatabase method.
4	openOrCreateDatabase(File file, SQLiteDatabase.CursorFactory factory) This method is similar to above method but it takes the File object as a path rather then a string. It is equivalent to file.getPath()

Database - Insertion

We can create table or insert data into table using execSQL method defined in SQLiteDatabase class. Its syntax is given below

```
mydatabase.execSQL("CREATE TABLE IF NOT EXISTS table_name(Username VARCHAR,Password VARCHAR);");

mydatabase.execSQL("INSERT INTO table_name VALUES('admin','admin');");
```

This will insert some values into our table in our database. Another method that also does the same job but take some additional parameter is given below

Sr.No	Method & Description
1	execSQL(String sql, Object[] bindArgs) This method not only insert data , but also used to update or modify already existing data in database using bind arguments

Database - Fetching

We can retrieve anything from database using an object of the Cursor class. We will call a method of this class called rawQuery and it will return a resultset with the cursor pointing to the table. We can move the cursor forward and retrieve the data.

```
Cursor resultSet = mydatbase.rawQuery("Select * from table_name",null);

resultSet.moveToFirst();

String username = resultSet.getString(0);

String password = resultSet.getString(1);
```

There are other functions available in the Cursor class that allows us to effectively retrieve the data. That includes

Sr.No	Method & Description
1	getColumnCount() This method return the total number of columns of the table.
2	getColumnIndex(String columnName) This method returns the index number of a column by specifying the name of the column
3	getColumnName(int columnIndex) This method returns the name of the column by specifying the index of the column
4	getColumnNames() This method returns the array of all the column names of the table.
5	getCount() This method returns the total number of rows in the cursor
6	getPosition() This method returns the current position of the cursor in the table
7	isClosed() This method returns true if the cursor is closed and return false otherwise

Database - Helper class

For managing all the operations related to the database, a helper class has been given and is called SQLiteOpenHelper. It automatically manages the creation and update of the database. Its syntax is given below

```
public class DBHelper extends SQLiteOpenHelper {

  public DBHelper(){

    super(context,DATABASE_NAME,null,1);

  }

  public void onCreate(SQLiteDatabase db) {}

  public void onUpgrade(SQLiteDatabase database, int oldVersion, int newVersion) {}}
```