MOBILE APPLICATION DEVELOPMENT – UNIT -1

- **Introduction:**
  - Introduction to mobile application development
  - Trends
  - Introduction to various platforms
  - Introduction to smart phones

Introduction to MAD

- **Introduction:**
- Mobile application development is the set of processes and procedures involved in writing software for small, wireless computing devices, such as smartphones and other hand-held devices.
- Mobile apps are often written specifically to take advantage of the unique features of a particular mobile device..
- For example a mobile health app might be written to take advantage of a smartwatch's temperature sensor.

Introduction contd…

- **Introduction:**
- Mobile application development is the process to making software for smartphones and digital assistants, most commonly for Android and iOS.
- The software can be preinstalled on the device, downloaded from a mobile app store or accessed through a mobile web browser.
- The programming and markup languages used for this kind of software development include Java, Swift, C# and HTML5.

Introduction contd…

Types of Mobile Applications:

- Native applications
- Progressive web apps
- Hybrid apps

Introduction contd…

- **Native Applications:**
- Native Apps are developed specifically for one platform, and can take full advantage of all the device features.
- **Native apps** are built for a **specific** operating system.
- A native app developed for iOS operating system won't work on Android devices and vice-versa.
- Native apps enable you to customize necessary features, but they can be more costly than other technologies.

Introduction contd…

- **Native Applications:**

- They are also built using the specific **Integrated Development Environment (IDE)** for the given operating systems.
- **Android Studio** for Android Apps and **XCode** for iOS Apps.
- IOS apps are written in Objective-C and Swift
- Android apps are written in Java or Kotlin.
- Windows and Blackberry apps are written in C++.

Introduction to MAD

Introduction to MAD

- Pros
  - Very fast and responsive because they are built for that specific platform.
  - Best performance.
  - More interactive, intuitive and run much smoother in terms of user input and output.
- Cons
  - More expensive to develop compared to cross-platform and web applications.
  - Require more time to develop as one application has to be written in different languages for different platforms.
  - Higher cost of maintenance and pushing out updates, due to multiple source code bases.

Introduction to MAD

- **Progressive web apps:**
- A PWA is a website that looks and behaves as if it is a mobile app. These applications are developed with web technologies.
- **PWA** also called as **Mobile web apps** are the web applications to render/deliver **pages on web browsers** running in mobile devices.
- Since these apps target browsers, they work on different mobile operating systems.

Introduction to MAD

- **Progressive web apps:**
- We can view a **mobile web app** on Android, iOS or Windows tablets and phone devices.
- Softwares' used to develop these applications are generally HTML, CSS, JavaScript, JQuery.

Introduction to MAD

- **Hybrid Applications:**
- **Hybrid apps** are a mixture of both native and mobile web apps.
- These are applications developed to be used across multiple platforms i.e can be deployed on both iOS and Android platforms.
- These are web apps that act like native apps.
- They are developed using technologies such as HTML, JavaScript and Cascading Style Sheets (CSS).

Introduction to MAD

- **Hybrid Applications:**
- Hybrid apps are more cost-effective to develop than native apps and can be created faster, but they aren't as feature-rich as native applications.
- Some popular **frameworks** for building Hybrid applications are
- **Ionic Framework**
- **PhoneGap**
- **Sencha Touch**
- **Apache Cordova**

Introduction to MAD

Trends

- Accelerated Mobile Pages (AMP)
  - AMP is a project undertaken by Google in association with Twitter and includes several other large search, social and web publishing platforms around the world.
  - AMP Project is an open-source website publishing technology designed to improve the performance of web content and advertisements.
  - As an alternative, the webpage can be displayed simultaneously while it is loading instead of making the user wait for the whole page to load - *it is* called *lazy loading*.

Trends

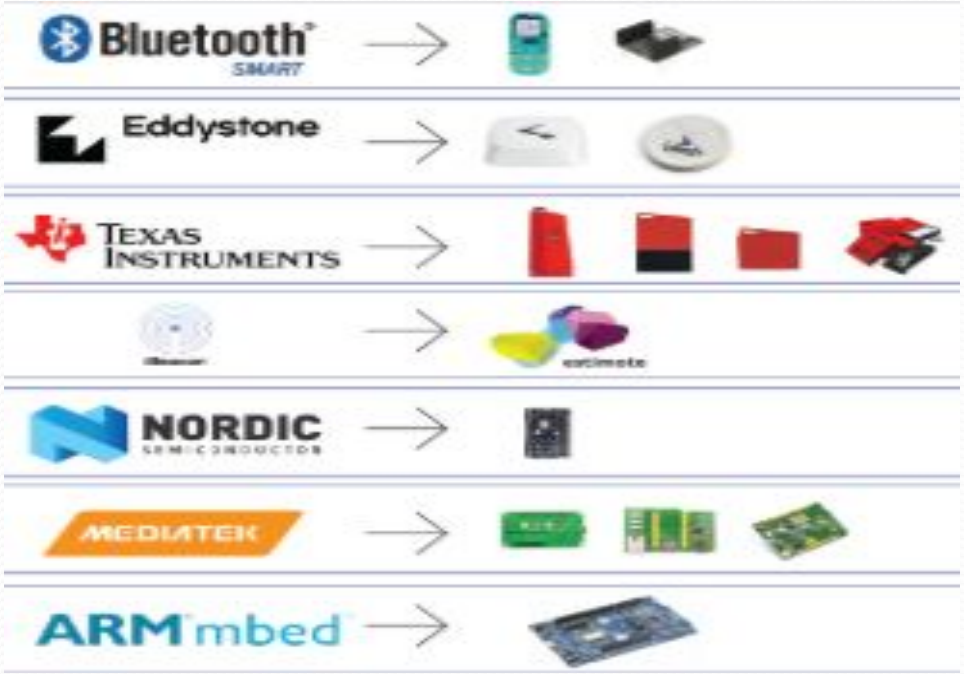
- AR & VR Apps
  - With is technology, one could feel like they are inside the imaginary world.
- Cloud-driven Mobile Apps
  - A cloud-driven or cloud application is simply an mobile application that runs on the server instead of your device.
  - A cloud app will serve same features and characteristics as the pure desktop or device app, but functions in the cloud storage.
- Enterprise Apps & Micro-apps
  - In the world of mobile, a micro app is a consumer-oriented application delivering highly targeted functionality (e.g. a weather forecast app).

Trends

- Security In Apps
  - Your phone probably has sensitive information on it, so it may be a good idea to install some security software.
    - Avast Mobile Security
    - AVG
    - Sophos Mobile Security
    - Avira Antivirus
- Swift Is Much Swifter Now
- Artificial Intelligence Is Real Now
- Wearable Devices

Trends

- IoT Apps



- Beacons & Location Based Services

VARIOUS PLATFORMS

VARIOUS PLATFORMS

MONOCROSS

An Open Source Cross-Platform Mobile Framework.

NativeScript

Xamarin

RhoMobile Suite

kony

VARIOUS PLATFORMS

Mobile Application Development Platforms Comparison

Platform	Programming Language	Price	Cross-platform Deployment	Ratings (G2)
alpha Anywhere	HTML, CSS, JavaScript	1. $99/mo 2. $399/mo 3. $750/mo	• iOS (iPhone, iPad, iPod Touch)	4.9 ★★★★½
Flutter	C, C++	Free	• iOS • Android • Web (in beta)	4.5 ★★★★½
mx mendix	HTML5, JAVA	1. Free 2. $1917/mo	• All platforms • Mobile apps are browser-based	4.4 ★★★★½
Xamarin	C#	$25/mo	• Android • iOS • Windows Phone • Windows Store apps	4.4 ★★★★½

VARIOUS PLATFORMS

IONIC	HTML, CSS, JavaScript	1. Free 2. $42/mo 3. $102/mo	• Mobile • Web • Desktop	4.3 ★★★★½
Sencha An Idera, Inc. Company	HTML, CSS, JavaScript	1. Free 2. $1999/yr 3. $2499/yr	• Android • iOS • Kindle • BlackBerry • Bada	4.1 ★★★★½
PhoneGap	HTML, CSS, JavaScript	1. Free 2. $12/mo 3. $30/mo 4. $90/mo	• iPhone • Android • Tizen • BlackBerry • Symbian • Palm • Bada	4 ★★★★☆
NativeScript	JavaScript, TypeScript	$19/mo	• iOS • Android	4 ★★★★☆
swiftic	JavaScript	1. $57/mo 2. $576/yr	• iOS (iPad, iPhone, iPod Touch)	3 ★★★☆☆

VARIOUS PLATFORMS

NativeScript	JavaScript, TypeScript	$19/mo	• iOS • Android	4 ★★★★☆
swiftic	JavaScript	1. $57/mo 2. $576/yr	• iOS (iPad, iPhone, iPod Touch)	3 ★★★☆☆
unity	C#	Free	• iOS • Android • PC • Mac • Desktop browser • Xbox 360 • PS3	4.4 ★★★★½
	Java, Swift, Objective-C	Free	• iOS • Android	4.3 ★★★★½

https://www.spaceotechnologies.com/blog/mobile-app-development-platforms/

VARIOUS PLATFORMS

References

https://www.spaceotechnologies.com/blog/mobile-app-development-platforms/

https://www.guru99.com/mobile-app-development-tools.html

Introduction to Smart Phones

- Phones used to be all about making calls, but now your mobile can do so much more.
- Touchscreen smartphones allows you to access the internet, use social media, get live news updates, play music and video, and much more.
- They almost universally use touchscreens for control, however, which can be a challenge for people new to the technology.
- First touch screen phone – IBM simon -1992- it has touch screen, portable, had a calculator, email, and could work on networks

https://georgiasouthern.libguides.com/c.php?g=612229&p=4545365#:~:text=The%20IBM%20Simon%20was%20the,and%20could%20work%20on%20networks.

Introduction to Smart Phones

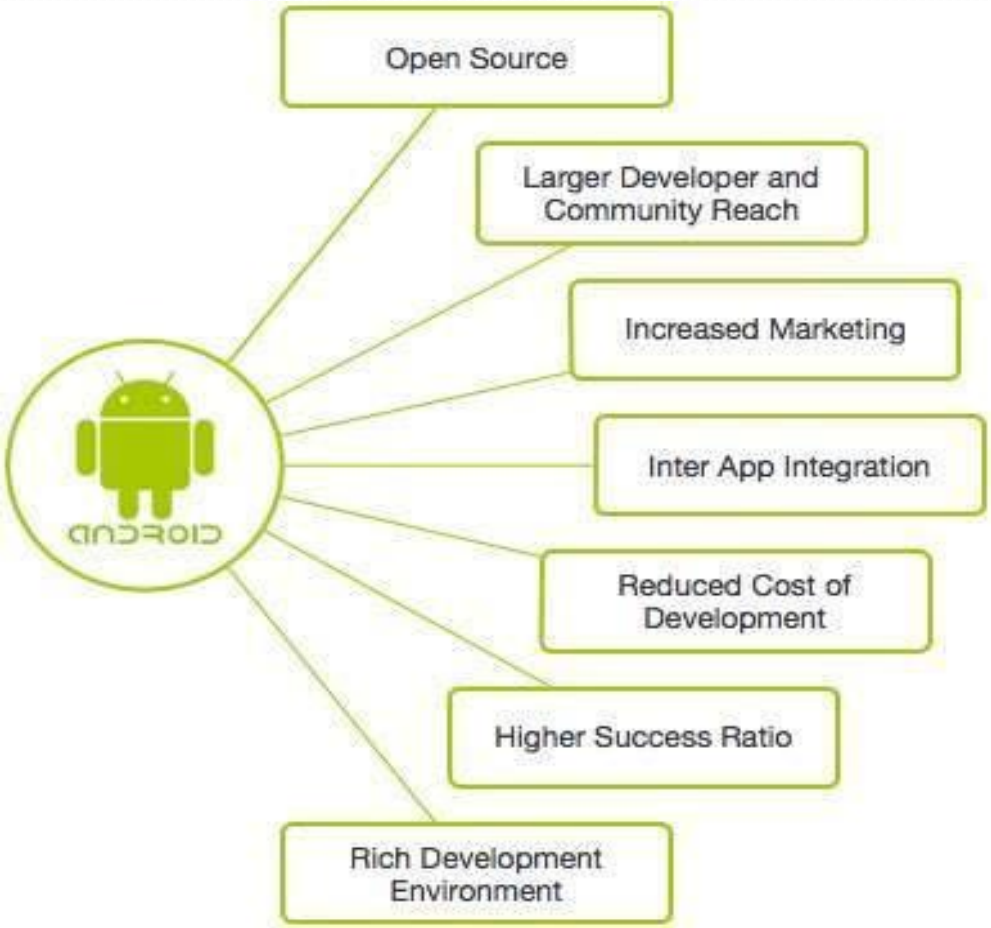
- **Android platform**:
  - Android platform features and architecture
  - Versions
  - Comparison added features in each versions
  - ART(Android Runtime)
  - ADB(Android Debug Bridge).

Android Platform

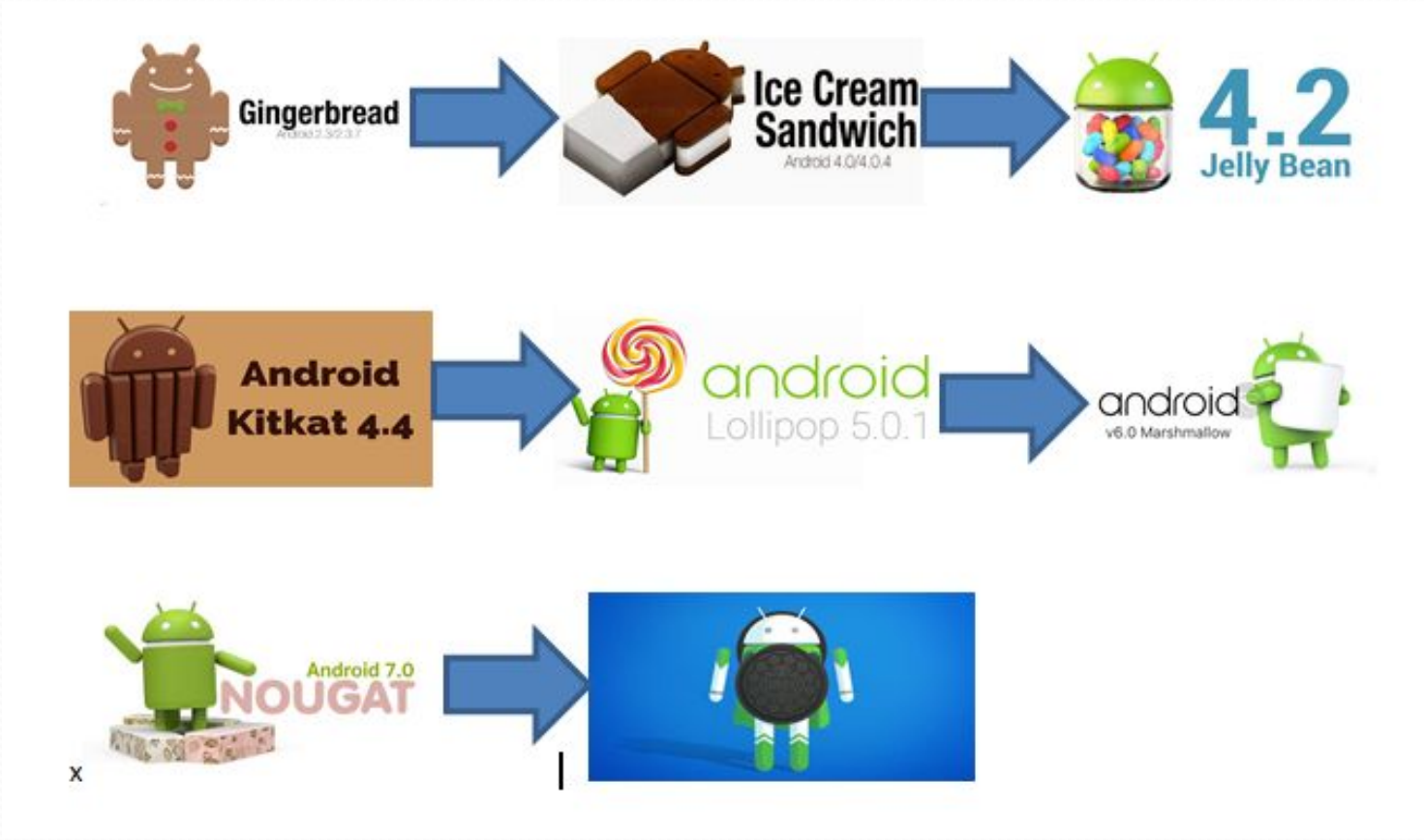
- **Android** is a software package and linux based operating system for mobile devices such as tablet computers and smartphones.
- developed by the ***Open Handset Alliance***, led by **Google**, and other companies.
- Java language is mainly used
- The goal of android project is to create a successful real-world product that improves the mobile experience for end users.

Android Platform

- **Why Android ?**



Android Platform



Android Features

- *Messaging*
  - SMS and MMS
- *Web browser*
  - Based on the open-source WebKit layout engine, coupled with Chrome's V8 JavaScript engine supporting HTML5 and CSS3.
- *Multi-touch*
  - Android has native support for multi-touch which was initially made available in handsets such as the HTC Hero.

Android Features

- *Multitasking*
  - User can jump from one task to another and same time various application can run simultaneously.
- *Multiple language support*
  - Supports single direction and bi-directional text.
- *Connectivity*
  - GSM/EDGE, IDEN, CDMA, EV-DO, UMTS, Bluetooth, Wi-Fi, LTE, NFC and WiMAX.

Android Features

- *Media support*
  - H.263, H.264, MPEG-4 SP, AMR, AMR-WB, AAC, HE-AAC, AAC 5.1, MP3, MIDI, Ogg Vorbis, WAV, JPEG, PNG, GIF, and BMP.
- *External storage*
  - SQLite, a lightweight relational database, is used for data storage purposes.
- Hardware support
- Bluetooth
- Tethering
- Streaming media support

Android Features

- *GCM*
  - Google Cloud Messaging (GCM) is a service that lets developers send short message data to their users on Android devices, without needing a proprietary sync solution
- *Wi-Fi Direct*
  - A technology that lets apps discover and pair directly, over a high-bandwidth peer-to-peer connection.
- *Android Beam*
  - A popular NFC-based technology that lets users instantly share, just by touching two NFC-enabled phones together

Android Features

- Auto Correction and Dictionary
- Voice-based features
- Screen capture
- TV recording
- Video calling
- Accessibility

Android Version and Comparison

	Version name	Key user features added	Key developer features added
Android 1.6	Donut	• Gesture framework • Turn-by-turn navigation	
Android 1.5	Cupcake	• Bluetooth A2DP, AVRCP support • Soft-keyboard with text-prediction • Record/watch videos	
Android 1.1	Banana bread	• "Show" & "Hide" numeric keyboard, in caller application • Ability to save MMS attachments	
Android 1.0	Apple pie	• Download and updates via Android Market • Web Browser • Camera support • Gmail, Contacts and Google Agenda synchronization • Google Maps • YouTube application	
Android 0.9	Coke		

	Version name	Key user features added	Key developer features added
Android 2.3.6	Gingerbread	• Voice search issue fixed	
Android 2.3.5	Gingerbread	• Improved network performance for the Nexus S 4G • Fixed Bluetooth issues on the Samsung Galaxy S • Gmail app. improvements	
Android 2.3.4	Gingerbread	• Voice or video chat using Google Talk	• Open Accessory API
Android 2.3.3	Gingerbread		• NFC API improvements (peer to peer communication...) • added unsecure bluetooth sockets
Android 2.3	Gingerbread	• Updated UI • Improved keyboard ease of use • Improved copy/paste • Improved power management • Social networking features • Near Field Communication support • Native VoIP/SIP support • Video call support	• performance - concurrent garbage collection, faster event distribution, updated video drivers • NDK - Native Asset Manager, Native Activities + event handling, khronos api • audio effects api • VP8, WebM, AAC, AMR wideband • Multiple camera sensor support • strictmode debugging • media framework replaces OpenCore
Android 2.2	Froyo	• Speed improvements • JIT implementation • USB Tethering • Applications installation to the expandable memory • Upload file support in the browser • Animated GIFs	
Android 2.1	Eclair	• Updated UI	
Android 2.0.1	Eclair		
Android 2.0	Eclair	• HTML • Digital zoom • Microsoft Exchange support • Bluetooth 2.1 • Live Wallpapers • Updated UI	

	Version name	Key user features added	Key developer features added
			apps
Android 3.2.6	Honeycomb	Minor fixes	
Android 3.2.4	Honeycomb	Added "Pay as you go" for tablets	
Android 3.2.2	Honeycomb	Minor fixes	
Android 3.2.1	Honeycomb	• Android Market updates including easier automatic updates • Google Books updates • Wi-Fi improvements • Chinese handwriting prediction improved	

Android 3.2	Honeycomb	• Optimizations for a wider range of tablets
		• Compatibility display mode (zoom for fixed-sized apps)
		• Media sync from SD card
			• Extended API for managing screens support
			  ◦ New resource qualifiers for screens support
			  ◦ New manifest attributes for screen-size compatibility
			  ◦ Screen compatibility mode which allows for phone apps to appear as if they were still on a phone

Android 3.1	Honeycomb	• UI improvements • Open Accessory API • USB host API • Mice, joysticks, gamepads... support • Resizable Home screen widgets • MTP notifications • RTP API for audio	

Android 3.0	Honeycomb	• Multi core support
		• Better tablet support
		• Updated 3D UI
		  ◦ customizable homescreens
		  ◦ recent applications viewing
		  ◦ redone keyboard layout
		• Media/Picture transport protocol
		• Google Talk video chat
		• Google eBooks
		• "Private browsing"
		• System-wide Clipboard
		• HTTP Live streaming
			• contextual action bar
			• Fragments first introduced(support library now supports it as well)
			• Hardware-accelerated 2D graphics
			• Renderscript 3D graphics engine
			• Pluggable DRM framework
			• device administration
			• High performance Animation Framework
			• RTP streaming API
			• Forced rendering of layers
			• High performance WIFI lock
			• Vew network traffic stats
			• ADTS AAC and FLAC audio
			• LRU cache

	Version name	Key user features added	Key developer features added
Android 4.3	Jelly Bean	• Dial pad auto-complete • Photo Sphere enhancements • Camera app UI updated • 4K resolution support • Ability to create restricted profiles for tablets • Hebrew and Arabic right-to-left (RTL) support • Bluetooth Low Energy (BLE) support • Bluetooth Audio/Video Remote Control Profile (AVRCP) 1.3 support • Security and performance enhancements	• OpenGL for Embedded Systems 3.0 graphics support • Logging and analyzing enhancements • Wi-Fi scanning API • Improved DRM (digital rights management) API • VP8 encoding
Android 4.2.2	Jelly Bean	• Allow toggling Wi-Fi and Bluetooth state in Quick Settings using long-press • Shows the percentage and estimated time remaining in the active download notifications • Wireless charging and low battery sounds changed • Gallery app updated for faster loading with new image transition • Performance enhancements and bug fixes (Bluetooth A2DP audio streaming fix...)	• Secure USB debugging (allow debugging to authenticated computers only)
Android 4.2.1	Jelly Bean	• Fix missing december bug in the People app • Add support for Bluetooth gamepads and joysticks HID devices	
Android 4.2	Jelly Bean	• Lockscreen widgets • 360 degree images with Photo Sphere • Gesture Typing, for faster typing • Wireless display with Miracast • Daydream to display information when idle or docked • Multi-user for tablets	• vsync timing • Triple buffering • reduced touch latency • CPU input boost • Native RTL support - mirrors the display from manifest prop • External display support - Display Manager • Nested fragments • Renderscript Compute - run tasks on the GPU (supported devices) • Renderscript ScriptGroups, built-in intrinsics like blur, • FilterScript is a subset of Renderscript made for high performance image processing

	Version name	Key user features added	Key developer features added
Android 7.1.2	Nougat	• Battery usage alerts • Nexus and Pixel specific improvements	
Android 7.1.1	Nougat	• Long press on the app icon enable new launch actions • The default keyboard allows now to send GIFs directly • New set of emojis	
Android 7.1	Nougat	• Daydream Virtual Reality mode • Night Light • Storage manager improvements • Performance improvements for Touch and Display managements • Option to enable fingerprint swipe down gesture • Seamless system updates	• Shortcut manager APIs • Support Circular app icons • Keyboard image insertion • VR thread scheduling improvements • Enhanced wallpaper metadata • Multi-endpoint call support • Source type support for Visual Voicemail • Carrier config options to manage video telephony
Android 7.0	Nougat	• Unicode 9.0 emoji • Better multitasking • Multi-window mode (PIP, Freeform window) • Seamless system updates (with dual system partition) • Better performance and code size thanks to new JIT Compiler	• Sustained Performance Mode (SPM) API • Vulkan 3D rendering API • Daydream virtual reality platform
Android 6.0.1	Marshmallow	New emojis	
Android 6	Marshmallow	• USB Type-C support • Fingerprint Authentication support • Better battery life with "deep sleep" • Permissions dashboard • Android Pay • MIDI support • Google Now improvements	• Custom Chrome Tabs for better in app browser support • App Permissions management update
Android 5.1.1	Lollipop	• Speed improvement • Bug fixes	
Android 5.1	Lollipop	• Multiple SIM cards support • Quick settings shortcuts to join Wi-Fi networks or control Bluetooth devices • Lock protection if lost or stolen • High Definition voice call • Stability and performance enhancements	
Android 5.0.2	Lollipop	• Performance improvements and bug fixes	

	Version name	Key user features added	Key developer features added
Android 10.1	Not Launched	• Not Launched New users features	• Not Launched New development features
Android 9.1	pie	• New Easter Egg • Bug fixes on UI • Support of Lowest Ram • New Boot Animation • Emoji updates (ie: Poop remove the mouth) • New Api speed improviment	• New Api development Style

Android 9	pie	• User interface updates:
		  ◦ Rounded corners across the UI
		  ◦ Quick settings menu change.
		  ◦ Notification bar, the clock has moved to the left.
		  ◦ The "dock" now has a semi-transparent background.
		  ◦ New transitions when switching between apps, or within apps
		  ◦ Volume slider updated
		• Richer messaging notifications: with full conversation, large images, smart replies
		• The power options now has a "screenshot" button
		• Biometric authentication can now be disabled only once	

Android 8.1	Oreo	• Show battery level in "Quick Settings" for devices connected via Bluetooth • Navigation buttons dim when not in use • UI updates to 'Power Off' and 'Restart' • Toast messages are now white in color with same existing transparency • Automatic light and dark themes • Emoji updates (ie: Hamburger move the cheese slice position)	• Neural networks API for artificial intelligence • Shared memory API • WallpaperColors API • Programmatic Safe Browsing actions • Autofill framework updates • Go Edition: lightweight Android distribution for devices with less than 1 GB of RAM
Android 8.0	Oreo	• PIP: Picture-in-Picture with resizable windows • Android Instant apps • Improved notifications system • Improved system settings	

Android Version and Comparison

Android 10	**Quince Tart or Android Q (API 29)**	· Support for foldable smartphones with flexible displays · Dark mode for eyes comfortability · Navigation control over gesture quicker and intuitive ever · Sound amplifier with more clear sound · Smart reply suggestions for all messaging apps · Live caption for media playing on a smartphone · Undo app removal · Better notification control with many options
Android 11	**Red Velvet Cake**	· Native screen recording · Muting notifications during video · Increase touch sensitivity · Notification History · Auto revoke app permissions · Revamped menu and screenshot shortcuts · New text selection mode from one app to another · Undoing recently cleared applications · Airplane mode doesn't kill Bluetooth anymore · Face Unlock will require you to open your eyes in pixel 4 · App pinning in the share menu · Improved notification conversation shades · Conversation bubbles and context-aware dark mode · Improved one-time permissions

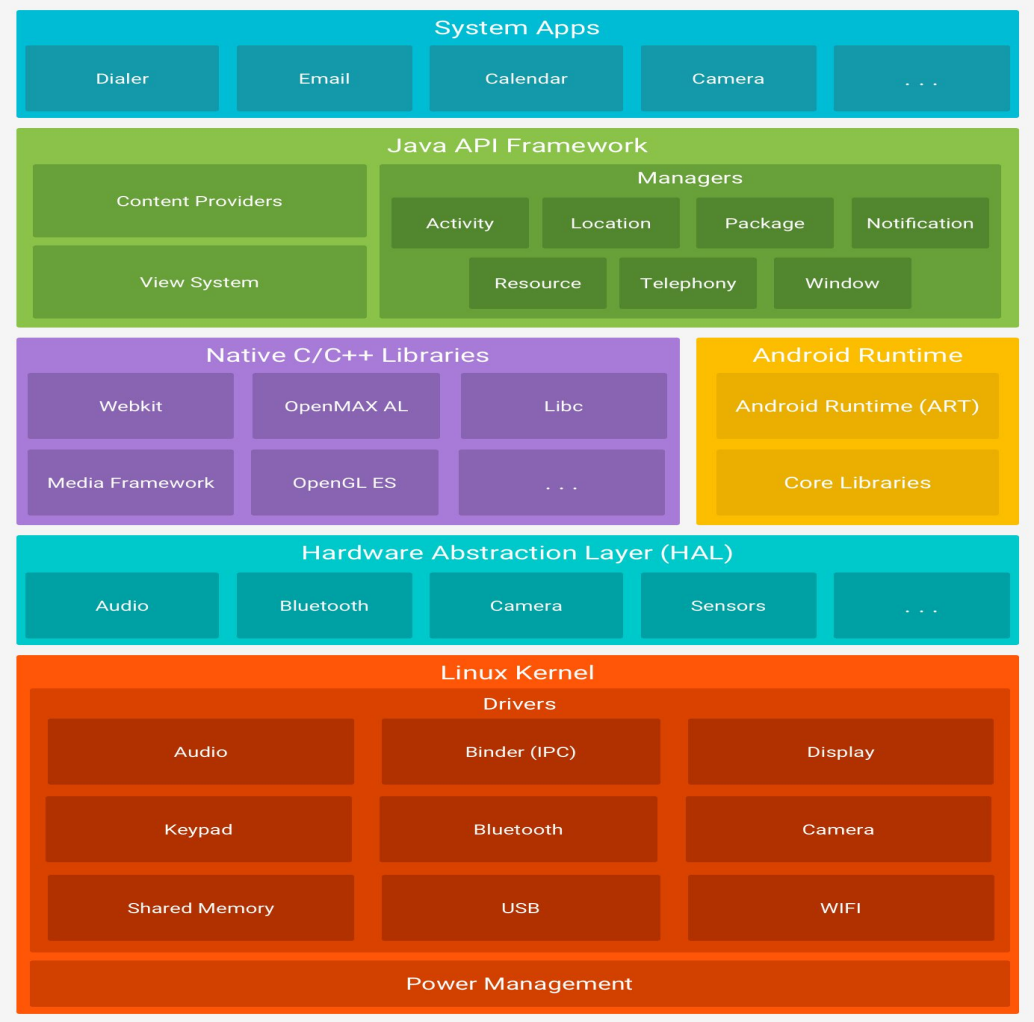
Android Version and Comparison

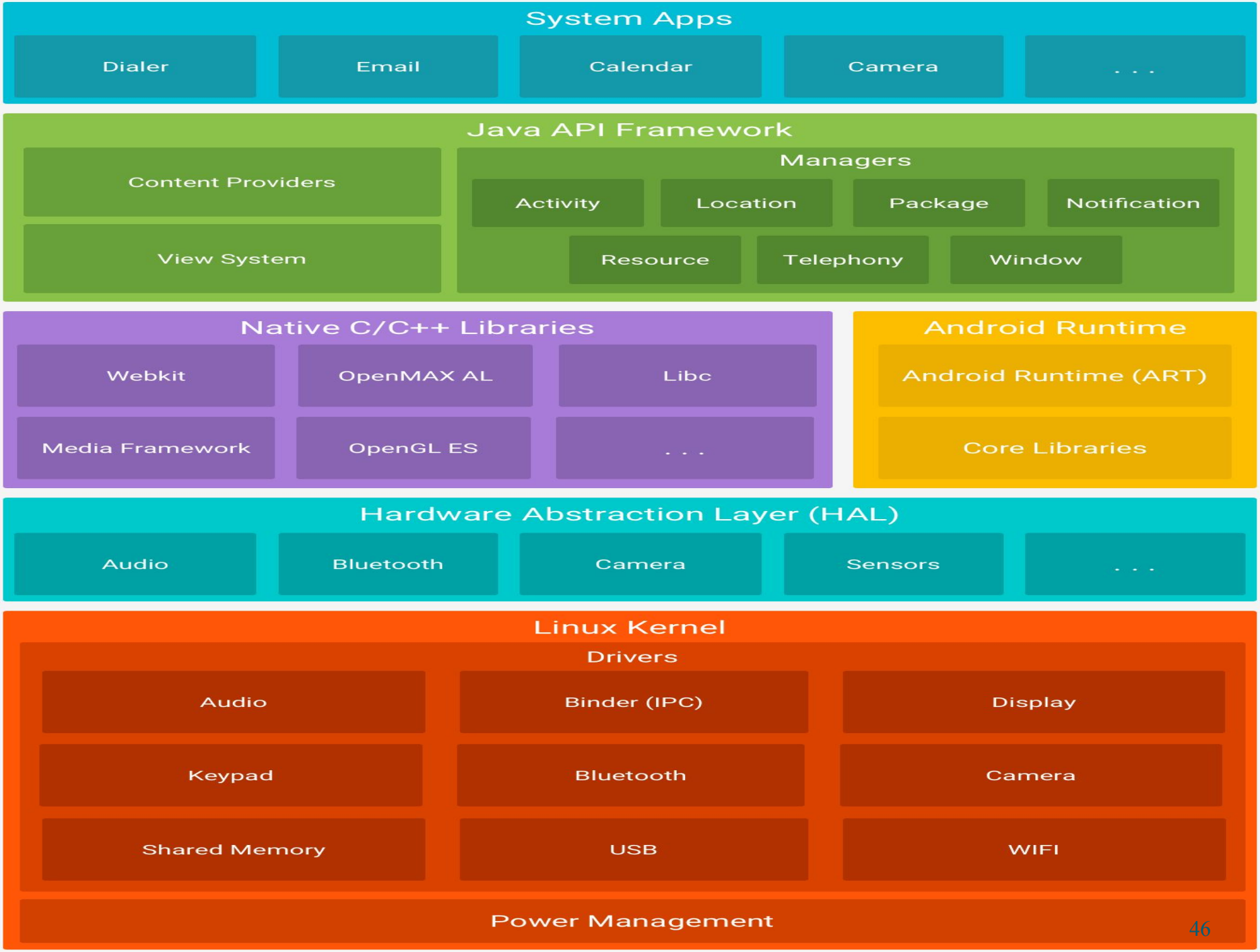
Android 12	**Snowcone**	· Dynamic colour. · Responsive motion. · Conversation widgets. · Accessibility improvements. · Mic & camera indicators and toggles. · Approximate location permissions. · Privacy dashboard. · Private Compute Core. · Effortless. · Enhanced gaming. · Scrolling screenshots
Android 13	**Tiramisu**	· Auto-theming icons · Alternative lock screen clock setup · Now Playing widget updated · Google Weekly · More intuitive QR scanner support · Native Bluetooth LE Audio support · Silent mode is truly silent · Tap to transfer media controls · Reworked audio output selector · 'Panlingual' per-app language settings · Secondary profiles for NFC payments · Privacy and security updates · NEARBY_WIFI_DEVICES

Android Version and Comparison

Android 14	**Upside Down Cake**	· Improved Battery Life · Larger Fonts and Smarter Scaling · Notification Flashes · Better Support for Large Screens · Restricting Photo and Video Access · Enhanced Security

ANDROID - ARCHITECTURE




System Apps
Dialer
Email
Calendar
Camera
. . .
Java API Framework
Content Providers
View System
Managers
Activity
Location
Package
Notification
Resource
Telephony
Window
Native C/C++ Libraries
Webkit
OpenMAX AL
Libc
Media Framework
OpenGL ES
. . .
Android Runtime
Android Runtime (ART)
Core Libraries
Hardware Abstraction Layer (HAL)
Audio
Bluetooth
Camera
Sensors
. . .
Linux Kernel
Drivers
Audio
Binder (IPC)
Display
Keypad
Bluetooth
Camera
Shared Memory
USB
WIFI
Power Management

The Linux Kernel

- The foundation of the Android platform is the *Linux kernel.*
- This provides a level of abstraction between the device hardware and it contains all the essential *hardware drivers like camera, keypad, display, networking etc*.
- Bridge between OS and Hardware

Handles…

- memory management, file management
- task scheduling, process management, managing hardware devices.

Hardware Abstraction Layer (HAL)

- The HAL provides is a logical division of code that serves as an abstraction layer between a computer's physical **hardware and its software**.
- It provides a device driver interface allowing a program to communicate with the hardware.
- When a framework API makes a call to access device hardware, the Android system loads the library module for that hardware component.

Libraries

- Libraries including Open-source *Web browser* engine **WebKit**, well known standard c library - **libc**, *SQLite database* which is a useful repository for *storage and sharing* of application data, **mediaframework** - libraries to *play and record audio and video, SSL libraries* responsible for Internet security etc..

- **OpenMAX AL**: standardized interface between an application and multimedia middleware.
- It provides abstractions for routines that are especially useful for processing of audio, video, and still images.

Libraries

- **OpenGL ES(Embedded Systems)** - programming interface for rendering 2D and 3D computer graphics used by video games.

ANDROID LIBRARIES

- **android.app –** Provides access to the application model and is the cornerstone of all Android applications.
- **android.content –** Facilitates content access, publishing and messaging between applications and application components.
- **android.database –** Used to access data published by content providers and includes SQLite database management classes.
- **android.opengl –** Graphics library - A Java interface to the OpenGL ES 3D graphics rendering API.

ANDROID LIBRARIES

- **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 LIBRARIES & Android Runtime

- **android.webkit** – A set of classes intended to allow web-browsing capabilities to be built into applications.
- ***Android Runtime***
  - This section provides a key component called ***Dalvik Virtual Machine*** which is a kind of Java Virtual Machine specially designed and optimized for Android.
  - The Dalvik VM makes use of Linux core features like memory management and multi-threading, which is native in the Java language.
  - ART performs the translation of the application's **bytecode into native instructions** that are later executed by the device's runtime environment.

Android Runtime

- ***Android Runtime***
  - From 4.4 version KitKat, there is the concept of ART as an alternative to DVM.
  - ART(Android Run Time) is a successor of DVM which uses the same bytecode and .dex files (but not .odex files), aiming at performance improvements transparent to the end-users.

JAVA API Framework

- ***Application Framework***
  - The Application Framework layer provides many **higher-level services** to applications in the form of Java classes.
  - **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.

JAVA API Framework & APPLICATIONS

- **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**
  - You will find all the Android application at the top layer. You will write your application to be installed on this layer only.
  - Examples of such applications are Contacts Books, Browser, Games etc.

Android Runtime (ART)

- Android Runtime (ART) is an application runtime environment used by the Android operating system.
- Replacing Dalvik Virtual Machine, the process virtual machine originally used by Android.
- ART performs the translation of the application's bytecode into native instructions that are later executed by the device's runtime environment.(ART introduced in Android L).
- ART is software layer between applications and operating system.
- It provide mechanism for executing java language.

Android Runtime (ART)

- ART perform two major things to achieve this
  - Runs Android framework and Applications using hybrid model of Interpreter, JIT and profile based Ahead of time compilation(AOT).
  - The Just-In-Time (JIT) compiler is a component of the Java Runtime Environment that improves the performance of Java applications at run time.
  - Memory Management using Memory allocator and Concurrent compacting Garbage collector.

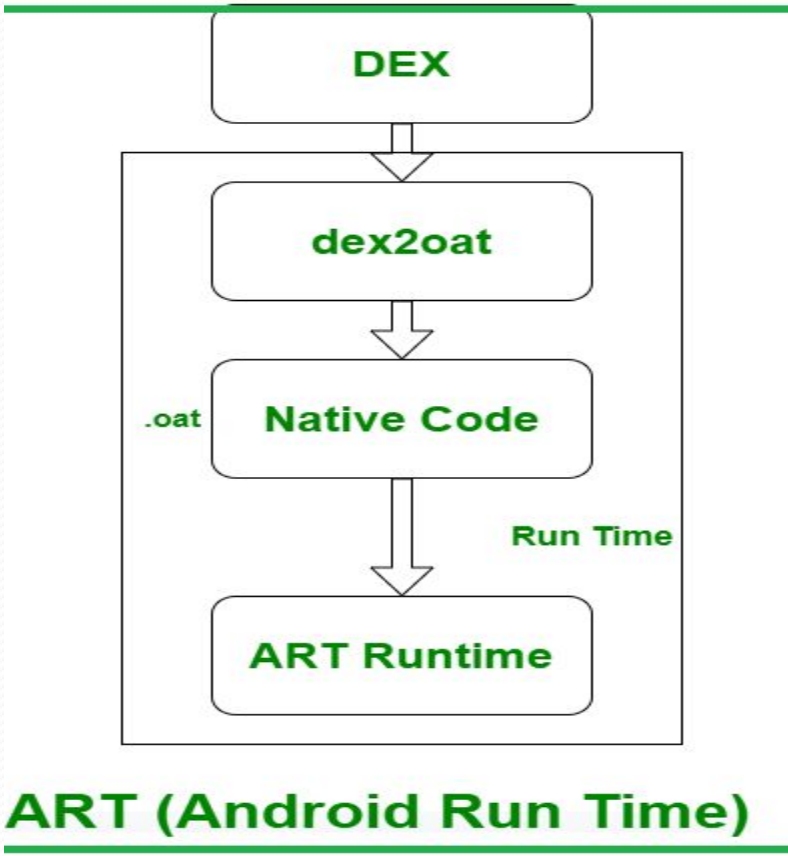
bytecodes into native machine code

Android Runtime (ART)

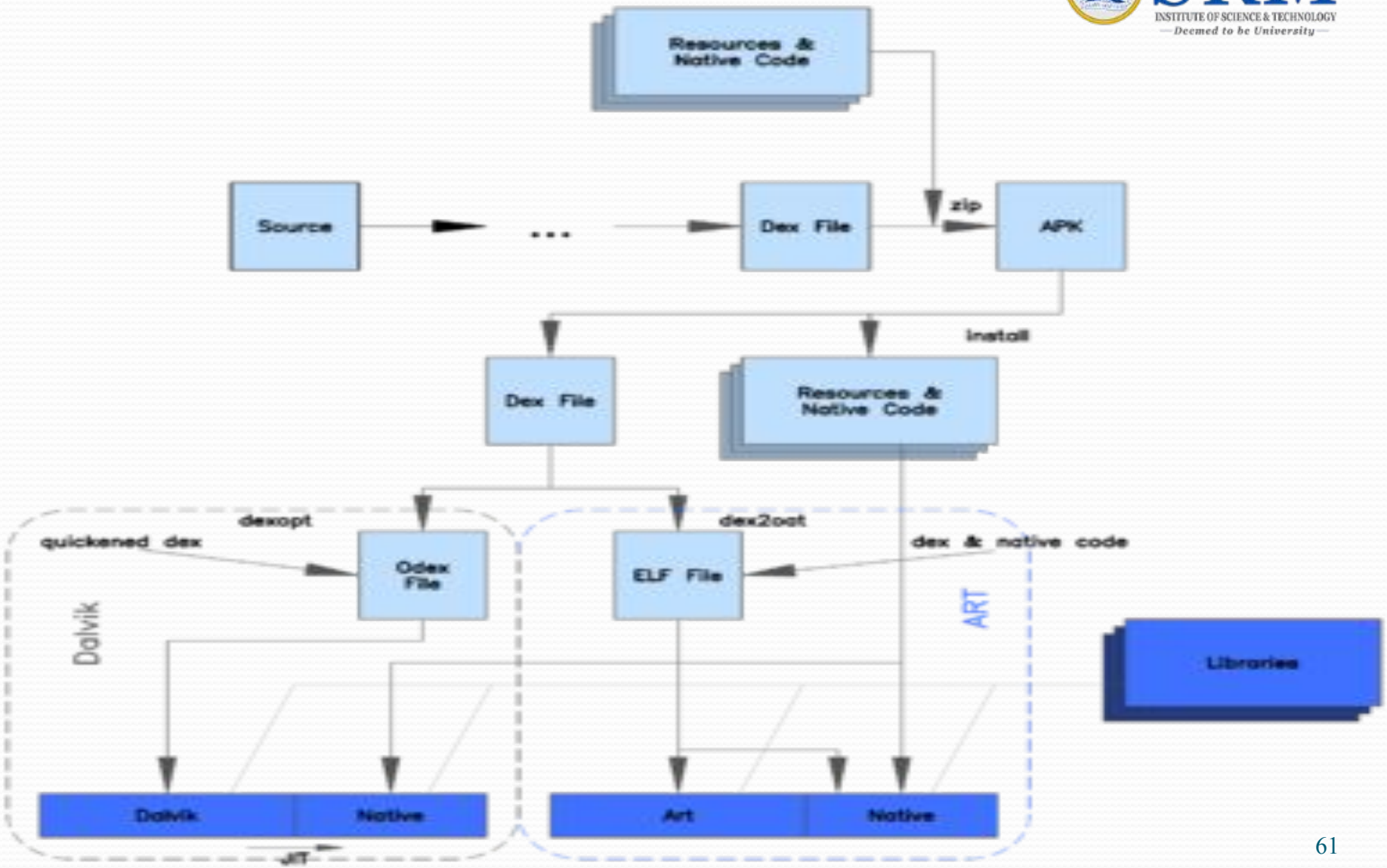
- ART(Android Run Time) replaces DVM from KitKat 4.4
- ART uses the same bytecode and .dex files (but not .odex files)
- **.dex(Dalvik Executable file)** file is an android's compiled code file. These .dex files are then **zipped** into a single .apk file.
- ART introduced aiming at improved performance.
- **DVM** converts bytecode every time you launch a specific app. But **ART** converts it just once during the installation of the app.

Android Runtime (ART)

- dex2oat is used to optimize and compile .dex **into** .oat file which contain machine code.
- ART compiles apps using the on-device dex2oat tool
- When an app is installed, Android automatically optimizes app data and **(dex2oat)**creates a OAT file.
- An OAT file is created by the Android operating system in order to speed up the loading time of an Android app (.APK file).



Android Runtime (ART)



Android Debug Bridge(ADB)

- Android Debug Bridge (adb) is a versatile command-line tool that lets you communicate with a device.
- The ADB command facilitates a variety of device actions, such as installing and debugging apps, and it provides access to a Unix shell that you can use to run a variety of commands on a device.
- It is a client-server program that includes three components:
  - *A client*, which sends commands. The client runs on your development machine. You can invoke a client from a command-line terminal by issuing an adb command.

Android Debug Bridge(ADB)

- *A daemon* ***(adbd),*** which runs commands on a device. The daemon runs as a background process on each device.
- *A server*, which manages communication between the client and the daemon. The server runs as a background process on your development machine.

- **Development environment/IDE**:
  - Android studio and its working environment
  - Gradle build system
  - Emulator setup

Development environment – IDE for Andriod

- **Android Studio**: Google's official IDE for Android based on Intellij IDEA
- **Eclipse**: This open source IDE is the second most popular in the world.
- **Visual Studio (with Xamarin)**: Microsoft's flagship IDE integrated with Xamarin , cross-platform native development. (Paid)
- **IntelliJ IDEA**: IntelliJ IDEA is described as a "capable and ergonomic IDE for JVM."(written in Java) (Paid)
- **NetBeans**: NetBeans is known primarily as a Java IDE, but it also supports many other languages. It is the official IDE for Java 8

Development environment – IDE for Andriod

- **Komodo**: IDE for Web and mobile app development by ActiveState (Paid)
- **Cordova**: First developed by a company called Nitobi, Cordova was rebranded as PhoneGap in 2011.
- **PhoneGap**: PhoneGap is Adobe's implementation of the Apache Cordova open source mobile development framework
- **Appcelerator Titanium**: open-source framework that used to create native mobile apps (Paid)
- **App Inventor**: App Inventor aims at helping students and other new developers create their first Android apps

Android Studio Features

- Android Studio offers even more features that enhance your productivity when building Android apps, such as:
  - A flexible *Gradle-based build system*
  - A fast and feature-rich emulator
  - A unified environment where you can develop for all Android devices
  - Instant Run to push changes to your running app without building a new APK
  - Code templates and GitHub integration to help you build common app features and import sample code
  - Extensive testing tools and frameworks.

Android Studio Features

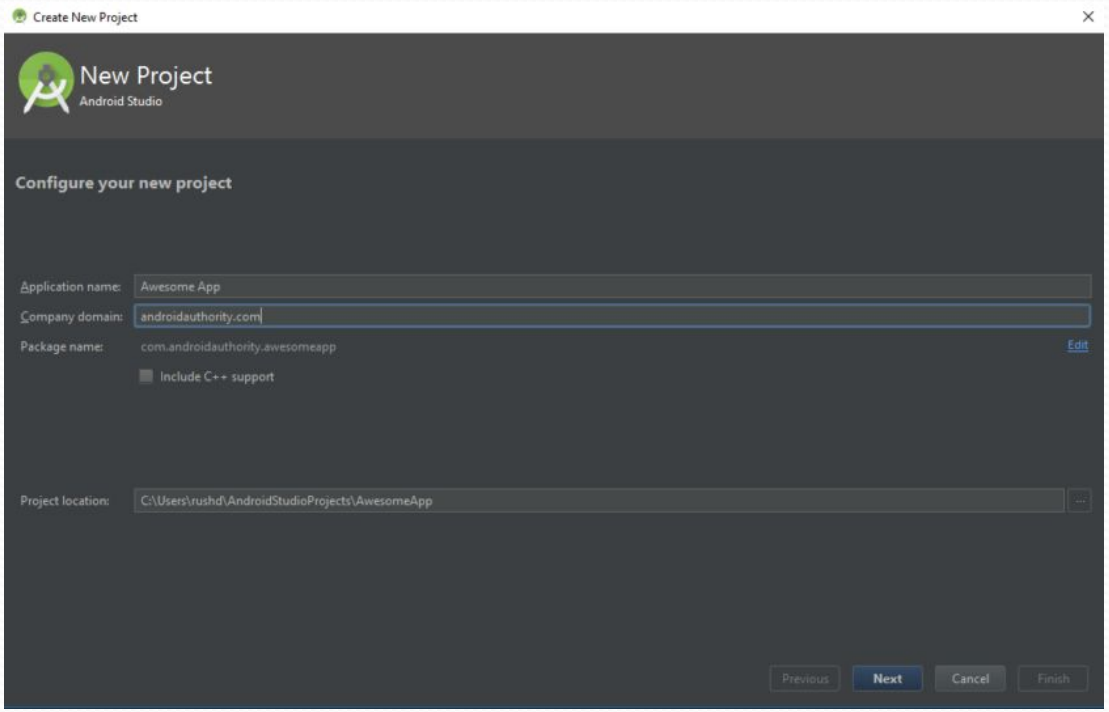
- Lint tools to catch performance, usability, version compatibility, and other problems
- C++ and NDK support
- Built-in support for Google Cloud Platform, making it easy to integrate Google Cloud Messaging and App Engine

Development environment/IDE

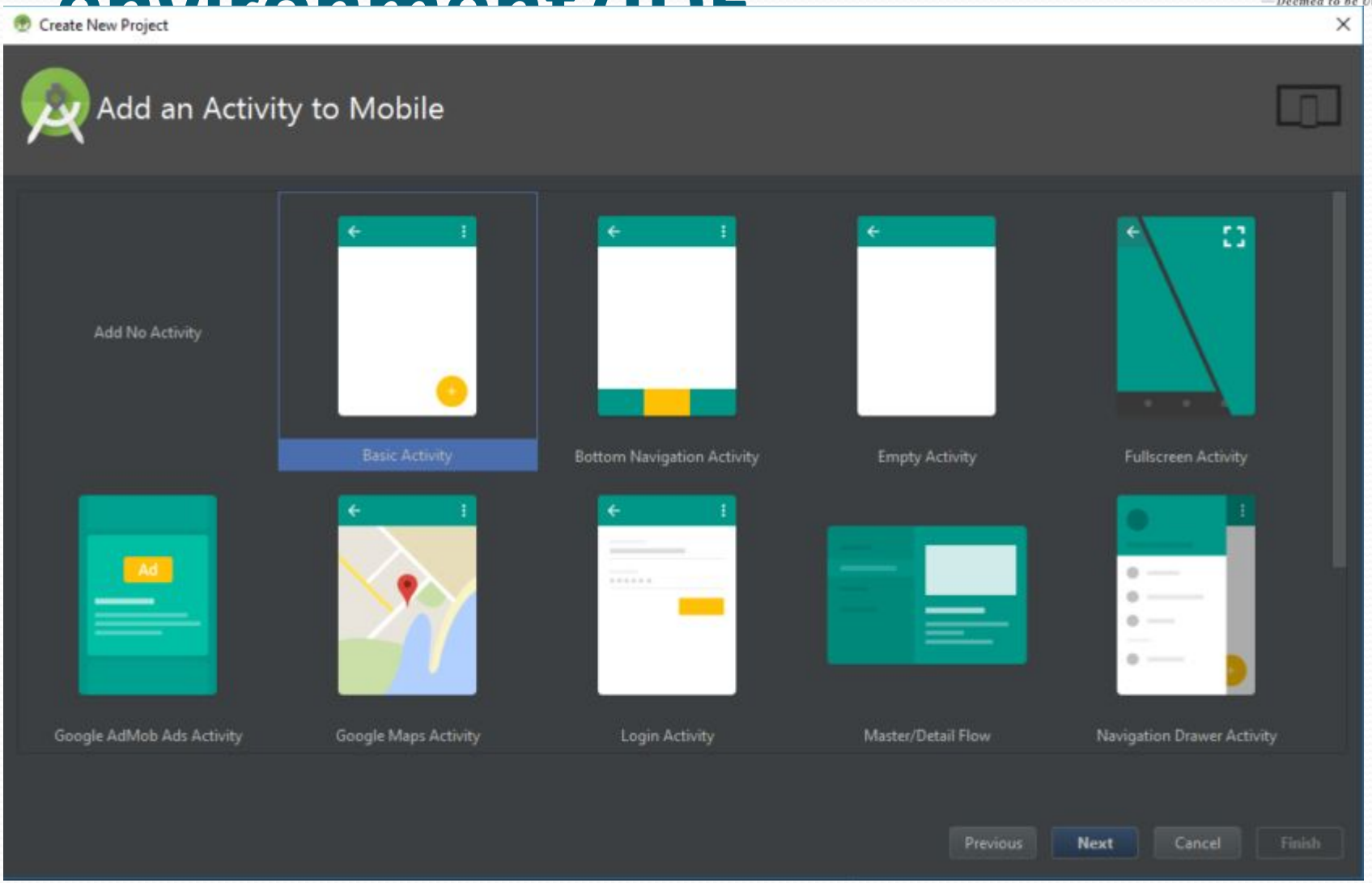
- Android application development on either of the following operating systems –
  - Microsoft Windows XP or later version.
  - Mac OS X 10.5.8 or later version with Intel chip.
  - Linux including GNU C Library 2.7 or later.
- Following is the list of software's needed before starting Android application programming.
  - Java JDK5 or later version
  - Android Studio

Development environment/IDE

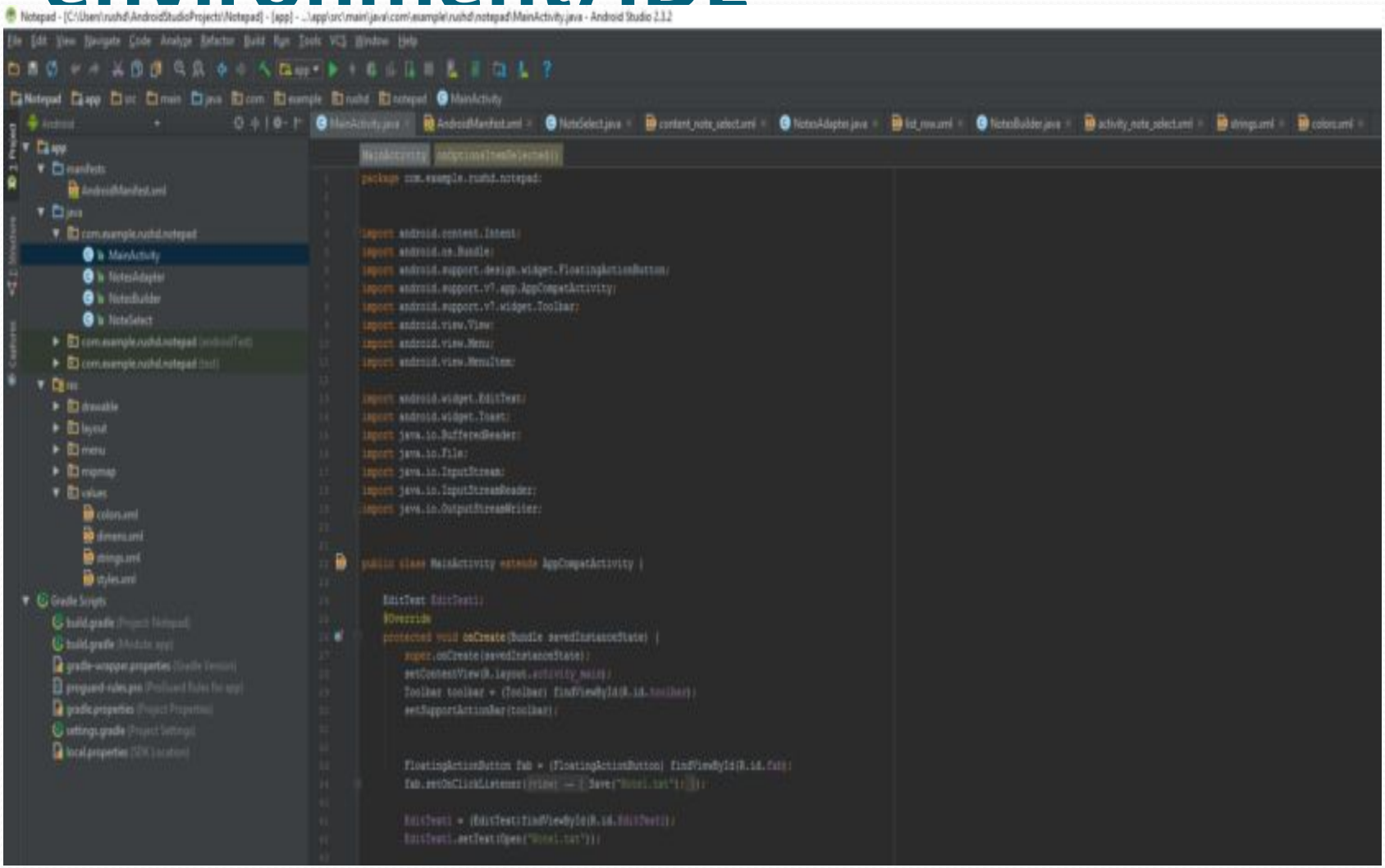
- Starting a new project
- Launching Android Studio and then selecting New Project, or you can choose *File > New > New Project* at any time from the IDE itself.



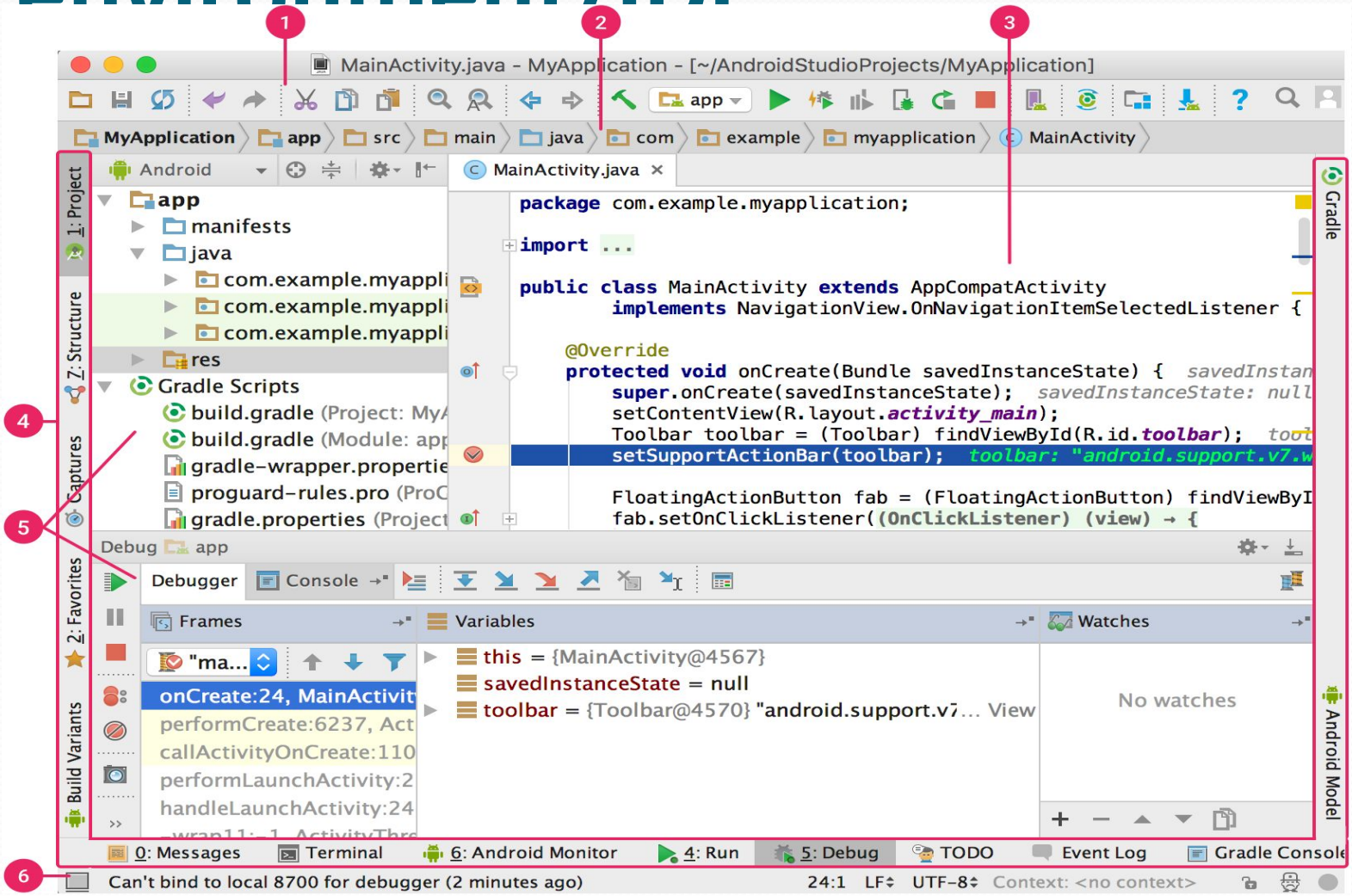
Development environment/IDE



Development environment/IDE



Development environment/IDE



Development environment/IDE

- *1. Toolbar* lets you carry out a wide range of actions, including running your app and launching Android tools.
- *2. Navigation bar* helps you navigate through your project and open files for editing. It provides a more compact view of the structure visible in the Project window.
- *3. Editor window* is where you create and modify code. Depending on the current file type, the editor can change. For example, when viewing a layout file, the editor displays the Layout Editor.
- *4. Tool window bar* runs around the outside of the IDE window and contains the buttons that allow you to expand or collapse individual tool windows.

Development environment/IDE

- *5. Tool windows* give you access to specific tasks like project management, search, version control, and more. You can expand them and collapse them.
- *6. Status bar* displays the status of your project and the IDE itself, as well as any warnings or messages.

Gradle Build System

- Gradle is a build system (open source) that is used to automate building, testing, deployment, etc.
- Every Android project needs a Gradle for generating an apk from the *.java* and *.xml* files in the project.
- Simply put, a gradle takes all the source files (java and XML) and applies appropriate tools, e.g., converts the java files into dex files and compresses all of them into a single file known as **apk** that is actually used..

Gradle Build System

- Android Studio uses Gradle as the foundation of the build system, with more Android-specific capabilities provided by the Android plugin for Gradle.
- This build system runs as an integrated tool from the Android Studio menu, and independently from the command line.
- You can use the features of the build system to do the following:
  - Customize, configure, and extend the build process.
  - Create multiple APKs for your app, with different features using the same project and modules.
  - Reuse code and resources across sourcesets.

Emulator Setup

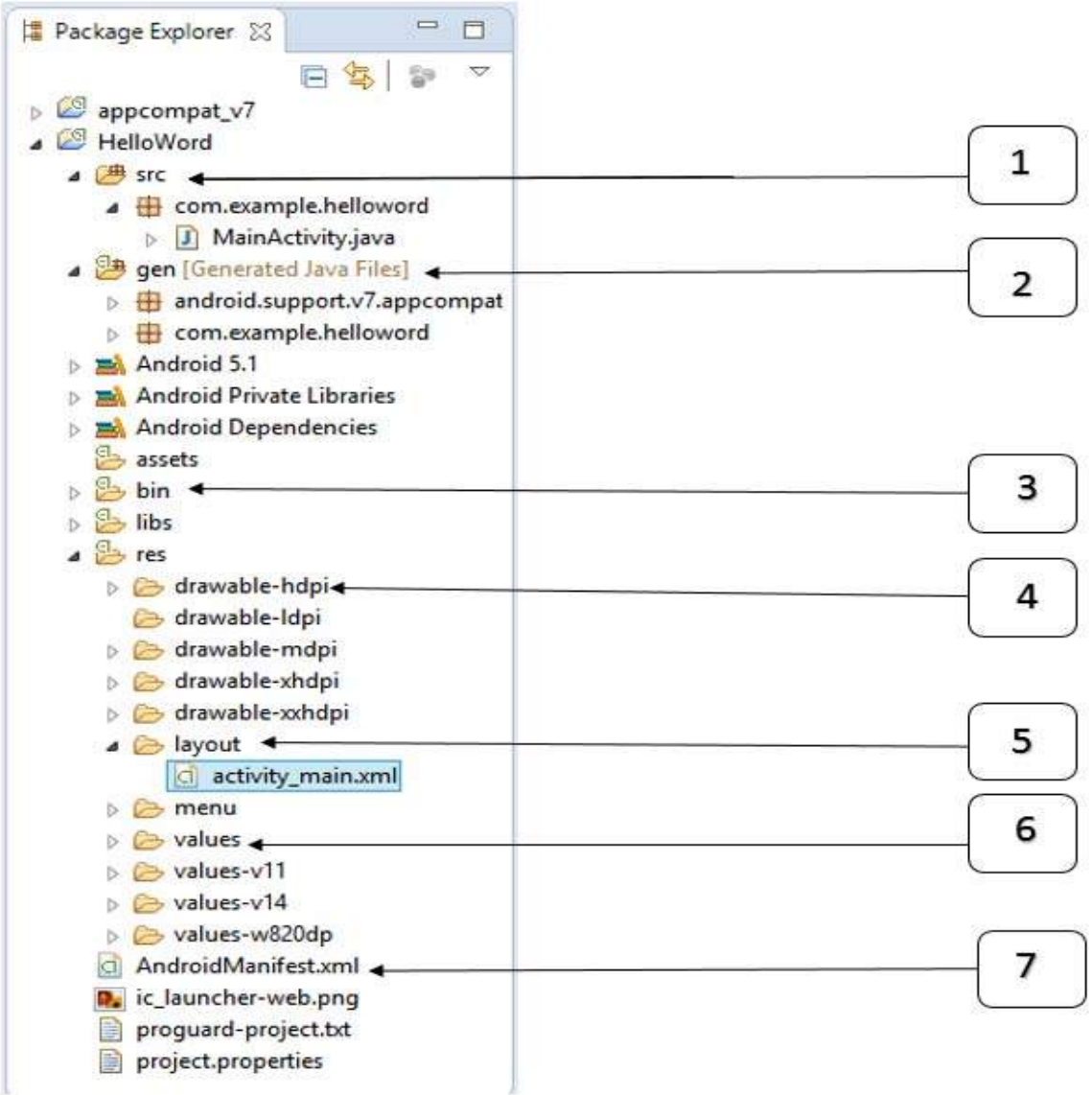
- The emulator lets you prototype, develop and test Android applications without using a physical device.
- Creating AVD
  - If you want to emulate a real device, first create an AVD with the same device configurations as real device, then launch this AVD from AVD manager.
- Changing Orientation
  - Usually by default when you launch the emulator, its orientation is vertical, but you can change it orientation by pressing Ctrl+F11 key from keyboard.

Emulator Commands

- Home - Shifts to main screen
- F2 - Toggles context sensitive menu
- F3 - Bring out call log
- F4 - End call
- F5 - Search
- F6 - Toggle trackball mode
- F7 - Power button
- F8 - Toggle data network
- Ctrl+F5 - Ring Volume up
- Ctrl+F6 - Ring Volume down

- **Application anatomy**:
  - Application framework basics: resources layout, values, asset XML representation and generated R.Java file, Android manifest file.
  - Creating a simple application.

Application anatomy:



Application anatomy:

1	**src** This contains the **.java** source files for your project. By default, it includes an **MainActivity.java** source file having an activity class that runs when your app is launched using the app icon.
2	**gen** This contains the **.R** file, a compiler-generated file that references all the resources found in your project. You should not modify this file.
3	**bin** This folder contains the Android package files **.apk** built during the build process and everything else needed to run an Android application.
4	**res/drawable-hdpi** This is a directory for drawable objects that are designed for high-density screens.
5	**res/layout** This is a directory for files that define your app's user interface.
6	**res/values** This is a directory for other various XML files that contain a collection of resources, such as strings and colours definitions.
7	**AndroidManifest.xml** This is the manifest file which describes the fundamental characteristics of the app and

Layout Attributes

- *android:paddingLeft* - This is the left padding filled for the layout.
- *android:paddingRight* - This is the right padding filled for the layout.
- *android:paddingTop* - This is the top padding filled for the layout.
- *android:paddingBottom* - This is the bottom padding filled for the layout.

Generated R.Javafile

- **R.java** file is the glue between the activity Java files like MainActivity.java and the resources like strings.xml. It is an automatically generated file and you should not modify the content of the R.java file
- R.java is neat. R.java is elegant. Due to its subversiveness (in a good way) its makings are implicit. It is fun to learn about Android.R.
- At a mile high level, every application has resouces. Familiar example of resources are strings, colors, and bitmaps. Instead of hard coding strings in an application one will use an id for a string.

Generated R.Javafile

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">hello</string>
    <string name="app_name">hello appname</string>
</resources>
```

This file will automatically update a "java" class in your root package called R.java

```
package com.ai.android.helloworld;

public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int icon=0x7f020000;
    }
    public static final class id {
        public static final int b1=0x7f050001;
        public static final int text1=0x7f050000;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {

        public static final int app_name=0x7f040001;
        public static final int app_name1=0x7f040003;

        public static final int hello=0x7f040000;
        public static final int hello1=0x7f040002;
    }
}
```

XML representation

- XML stands for Extensible Mark-up Language.
- XML is a very popular format and commonly used for sharing data on the internet.
- This chapter explains how to parse the XML file and extract necessary information from it.
- Android provides three types of XML parsers which are **DOM,SAX and XMLPullParser**.
- Among all of them android recommend XMLPullParser because it is efficient and easy to use.
- So we are going to use XMLPullParser for parsing XML.

XML representation

- The first step is to identify the fields in the XML data in which you are interested in.
- For example. In the XML given below we interested in getting temperature only.

```xml
<?xml version="1.0"?>
<current>

   <city id="2643743" name="London">
      <coord lon="-0.12574" lat="51.50853"/>
      <country>GB</country>
      <sun rise="2013-10-08T06:13:56" set="2013-10-08T17:21:45"/>
   </city>

   <temperature value="289.54" min="289.15" max="290.15" unit="kelvin"/>
   <humidity value="77" unit="%"/>
   <pressure value="1025" unit="hPa"/>
</current>
```

XML Elements

- *Prolog* - An XML file starts with a prolog. The first line that contains the information about a file is prolog
- *Events* - An XML file has many events. Event could be like this. Document starts , Document ends, Tag start , Tag end and Text e.t.
- *Text* - Apart from tags and events, and xml file also contains simple text. Such as GB is a text in the country tag.
- *Attributes* - Attributes are the additional properties of a tag such as value e.t.c

ANDROID MANIFEST FILE

- Every app project must have an *AndroidManifest.xml*
- The manifest file describes essential information about your app such as **activities, services, broadcast receivers, content providers** etc.
- This file works as an interface between Android OS and your application
- It performs some other tasks also:
  - It is responsible to protect the application to access any protected parts by providing the permissions.
  - It also declares the *android api* that the application is going to use.
  - It lists the instrumentation classes. The instrumentation classes provides profiling and other information. These information are removed just before the application is published etc.

ANDROID MANIFEST FILE

- **<manifest** xmlns:android="http://schemas.android.com/apk/res/android"
- package="com.javatpoint.hello"
- android:versionCode="1"
- android:versionName="1.0" **>**
-
- **<uses-sdk**
- android:minSdkVersion="8"
- android:targetSdkVersion="15" **/>**
-
- **<application**
- android:icon="@drawable/ic_launcher"
- android:label="@string/app_name"
- android:theme="@style/AppTheme" **>**

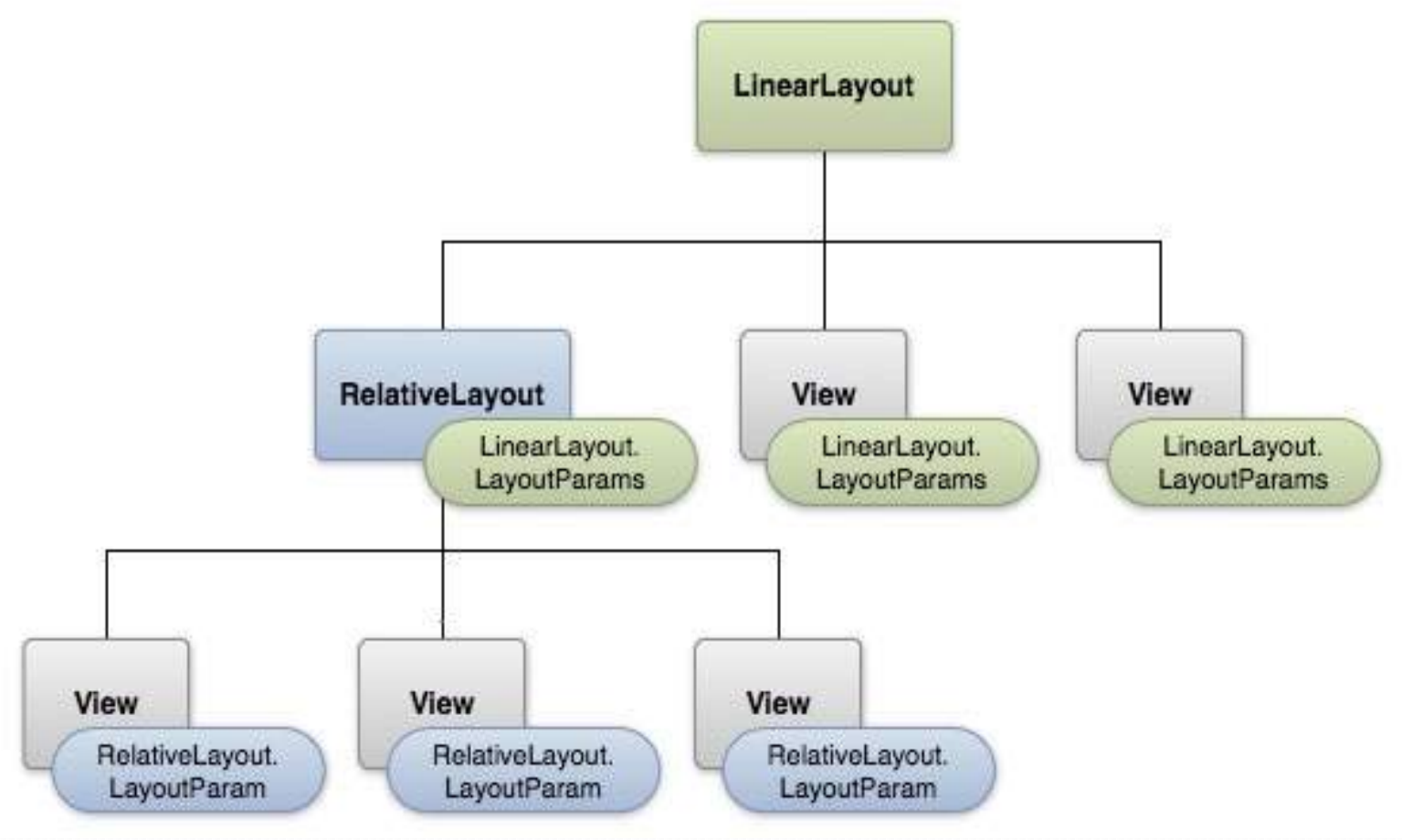
ANDROID MANIFEST FILE

```
        <activity
          android:name=".MainActivity"
          android:label="@string/title_activity_main" >
          <intent-filter>
            <action android:name="android.intent.action.MAIN" />

        <category android:name="android.intent.category.LAUNCHER" /
>
          </intent-filter>
        </activity>
    </application>

</manifest>
```

Android Layout

- The **activity_main.xml** is a layout file available in **res/layout** directory, that is referenced by your application when building its interface.
- You will modify this file very frequently to change the layout of your application.

Android - UI Layouts



Android - UI Layouts

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:orientation="vertical" >

   <TextView android:id="@+id/text"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="This is a TextView" />

   <Button android:id="@+id/button"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="This is a Button" />

   <!-- More GUI components go here  -->

</LinearLayout>
```

Android Layout Types

- *Linear Layout* is a view group that aligns all children in a single direction, vertically or horizontally.
- *Relative Layout* is a view group that displays child views in relative positions.
- *Table Layout* is a view that groups views into rows and columns.
- *Absolute Layout* enables you to specify the exact location of its children.
- *Frame Layout* is a placeholder on screen that you can use to display a single view.

Android Layout Types

- *List View* is a view group that displays a list of scrollable items.
- *Grid View* is a View Group that displays items in a two-dimensional, scrollable grid.

Layout Attributes

- *android:id* - This is the ID which uniquely identifies the view.
- *android:layout_width* - This is the width of the layout.
- *android:layout_height* - This is the height of the layout
- *android:layout_marginTop* - This is the extra space on the top side of the layout.
- *android:layout_marginBottom* - This is the extra space on the bottom side of the layout.
- *android:layout_marginLeft* - This is the extra space on the left side of the layout.
- *android:layout_marginRight* - This is the extra space on the right side of the layout.

Layout Attributes

- *android:layout_gravity* - This specifies how child Views are positioned.
- *android:layout_weight* - This specifies how much of the extra space in the layout should be allocated to the View.
- *android:layout_x* - This specifies the x-coordinate of the layout.
- *android:layout_y* - This specifies the y-coordinate of the layout.
- *android:layout_width* - This is the width of the layout.
- *android:layout_width* - This is the width of the layout.