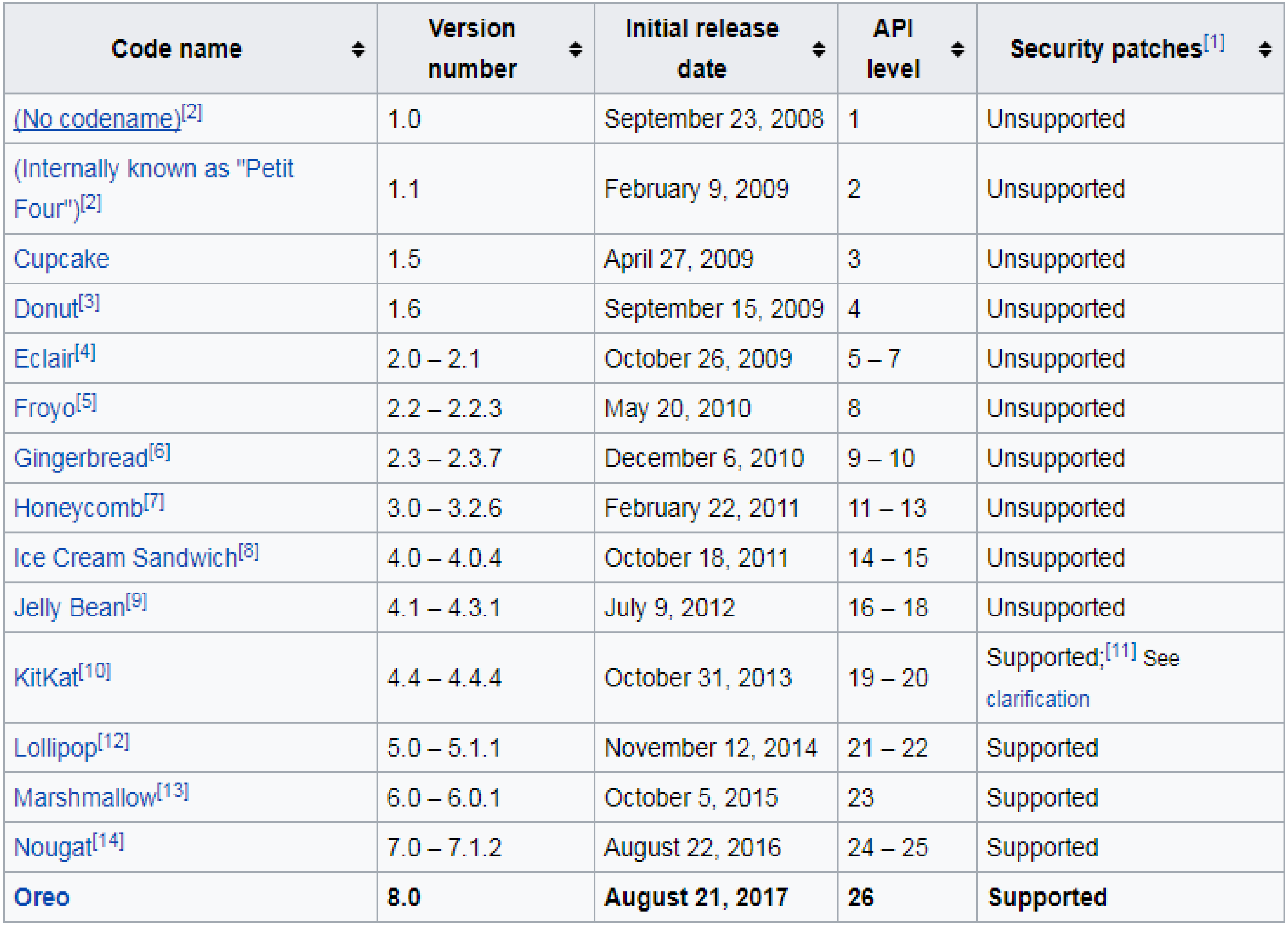
History

- The first beta version of the Android Software Development Kit (SDK) was released by Google in 2007, whereas the first commercial version, Android 1.0,was released in September 2008.
- On June 27, 2012, at the Google I/O conference, Google announced the next Android version, 4.1 **Jelly Bean**. Jelly Bean is an incremental update, with the primary aim of improving the user interface, both in terms of functionality and Performance.
- The source code for Android is available under free and open source software licenses. Google publishes most of the code under the Apache License version 2.0 and the rest, Linux kernel changes, under the GNU General Public License version 2.



Code name	Version number	Initial release date	API level	Security patches[1]
(No codename)[2]	1.0	September 23, 2008	1	Unsupported
(Internally known as "Petit Four")[2]	1.1	February 9, 2009	2	Unsupported
Cupcake	1.5	April 27, 2009	3	Unsupported
Donut[3]	1.6	September 15, 2009	4	Unsupported
Eclair[4]	2.0 – 2.1	October 26, 2009	5 – 7	Unsupported
Froyo[5]	2.2 – 2.2.3	May 20, 2010	8	Unsupported
Gingerbread[6]	2.3 – 2.3.7	December 6, 2010	9 – 10	Unsupported
Honeycomb[7]	3.0 – 3.2.6	February 22, 2011	11 – 13	Unsupported
Ice Cream Sandwich[8]	4.0 – 4.0.4	October 18, 2011	14 – 15	Unsupported
Jelly Bean[9]	4.1 – 4.3.1	July 9, 2012	16 – 18	Unsupported
KitKat[10]	4.4 – 4.4.4	October 31, 2013	19 – 20	Supported;[11] See clarification
Lollipop[12]	5.0 – 5.1.1	November 12, 2014	21 – 22	Supported
Marshmallow[13]	6.0 – 6.0.1	October 5, 2015	23	Supported
Nougat[14]	7.0 – 7.1.2	August 22, 2016	24 – 25	Supported
Oreo	**8.0**	**August 21, 2017**	**26**	**Supported**

Android version : Features

- https://en.wikipedia.org/wiki/Android_version_history

How to Install

- Eclipse + ADT bundle

  Dowload ADT bundle

  http://void-mainblog.blogspot.in/2015/04/download-eclipse-adt-bundle-for-android.html

- Android Studio

  https://developer.android.com/studio/index.html

ADT

- The ADT Bundle provides everything you need to start developing apps, including a version of the Eclipse IDE with built-in ADT (Android Developer Tools) to streamline your Android app development.
- To add the ADT plugin to Eclipse:

  --> Start Eclipse , then select Help > Install New Software .

  --> Click Add , in the top-right corner

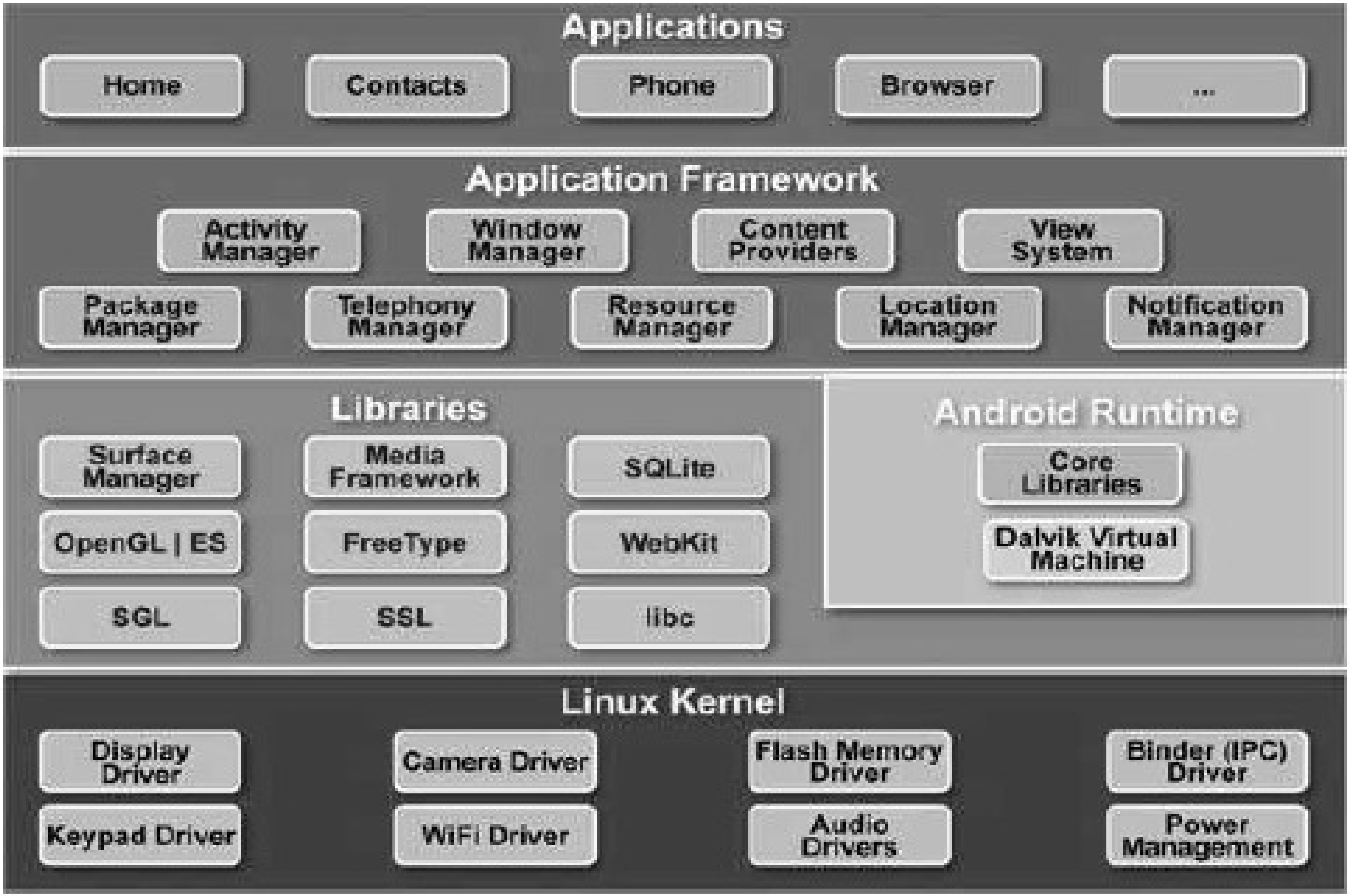
Android Features

- Beautiful UI
- Connectivity
- Storage
- Media
- Messaging
- Web browser
- Multi-touch
- Multi-tasking
- Resizable widgets
- Multi-Language

To Setup environment

- JDK
- Eclispe
- SDK (software development kit)
- ADT (Android development kit)

Architecture



Linux Kernel

At the bottom of the layers is Linux - Linux 2.6 with

- Process managment
- Memory managment
- Device managment

Libraries

On top of Linux kernel there is a set of libraries including open-source Web browser engine WebKit, well known library libc, SQLite database which is a useful repository for storage and sharing of application data, libraries to play and record audio and video, SSL libraries responsible for Internet security etc.

Android Runtime

Dalvik Virtual Machine which is a kind of Java Virtual Machine specially designed and optimized for Android.

Dalvik VM makes use of Linux core features like memory management and multi-threading, which is intrinsic in the Java language. The Dalvik VM enables every Android application to run in its own process, with its own instance of the Dalvik virtual machine.

The Android runtime also provides a set of **core libraries** which enable Android application developers to write Android applications using standard Java programming language.

DVM & ART

- ART (Android RunTime) is the next version of Dalvik. Dalvik is the runtime, bytecode, and VM used by the Android system for running Android applications. ART has two main features compared to Dalvik: Ahead-of-Time (AOT) compilation, which improves speed (particularly startup time) and reduces memory footprint (no JIT)

- Android runtime (ART) is the managed runtime used by applications and some system services on Android. ART and its predecessor Dalvik were originally created specifically for the Android project. ART as the runtime executes the Dalvik Executable format and Dex bytecode specification.

ART Features

- Ahead-of-time

  -->ART introduces ahead-of-time (AOT) compilation, which can improve app performance. ART also has tighter install-time verification than Dalvik.

  --> At install time, ART compiles apps using the on-device dex2oat tool. This utility accepts DEX files as input and generates a compiled app executable for the target device.

ART Features

- Improved GC

  Garbage collection (GC) can impair an app's performance, resulting in choppy display, poor UI responsiveness, and other problems

- Development and debugging improvements

ART Features

- Support for sampling profiler
- Historically, developers have used the Traceview tool (designed for tracing application execution) as a profiler. While Traceview gives useful information, its results on Dalvik have been skewed by the per-method-call overhead, and use of the tool noticeably affects run time performance.

- ART adds support for a dedicated sampling profiler that does not have these limitations. This gives a more accurate view of app execution without significant slowdown. Sampling support was added to Traceview for Dalvik in the KitKat release.

ART Features

- Support for more debugging features

  -- See what locks are held in stack traces, then jump to the thread that holds a lock.

  -- Ask how many live instances there are of a given class, ask to see the instances, and see what references are keeping an object live.

  -- Filter events (like breakpoint) for a specific instance.

  -- See the value returned by a method when it exits (using “method-exit” events).

  -- Set field watchpoint to suspend the execution of a program when a specific field is accessed and/or modified.

Application Framework

The Application Framework layer provides many higher-level services to applications in the form of Java classes. Application developers are allowed to make use of these services in their applications.

Applications

--All the Android application at the top layer.

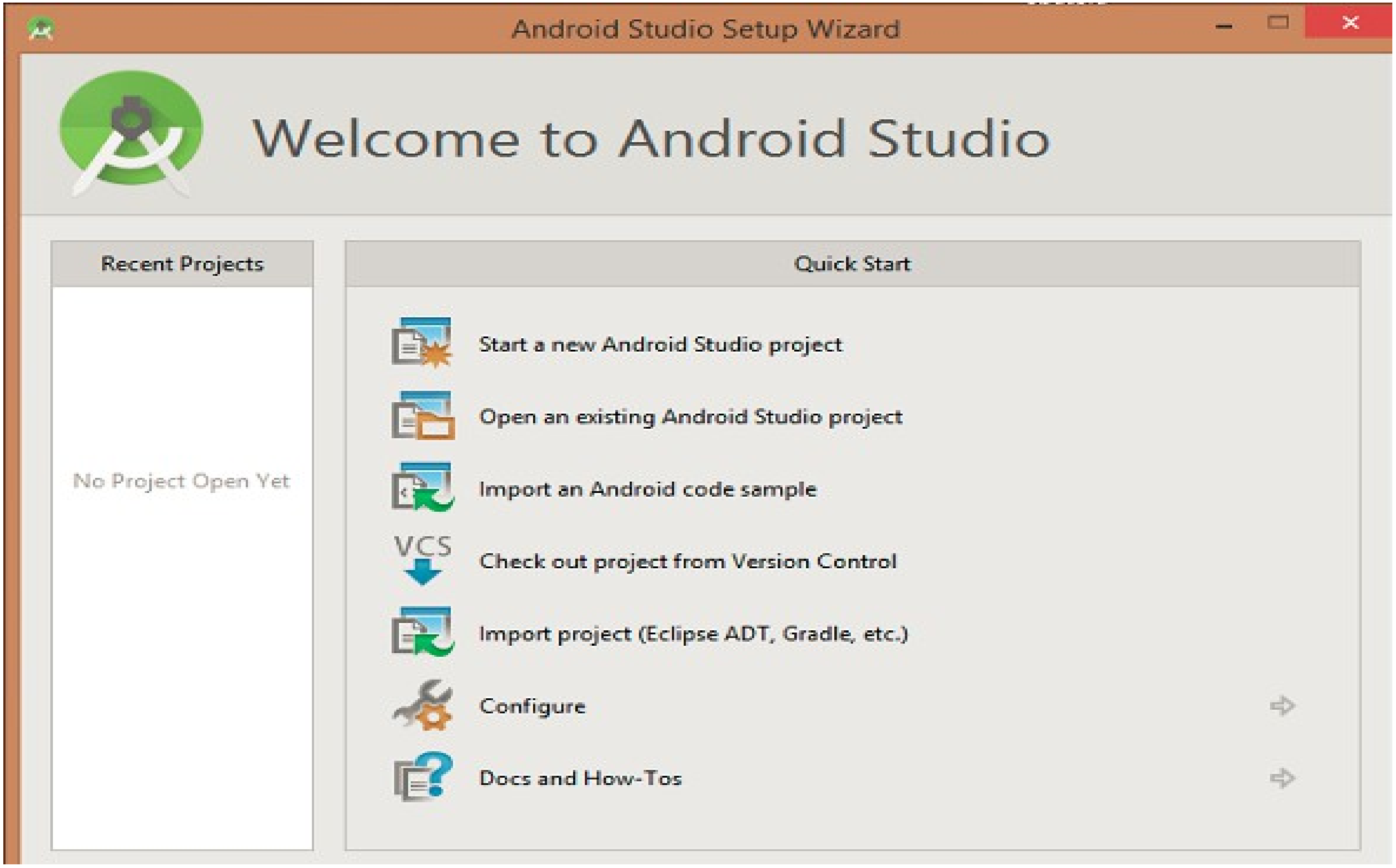
Basic Components of Android

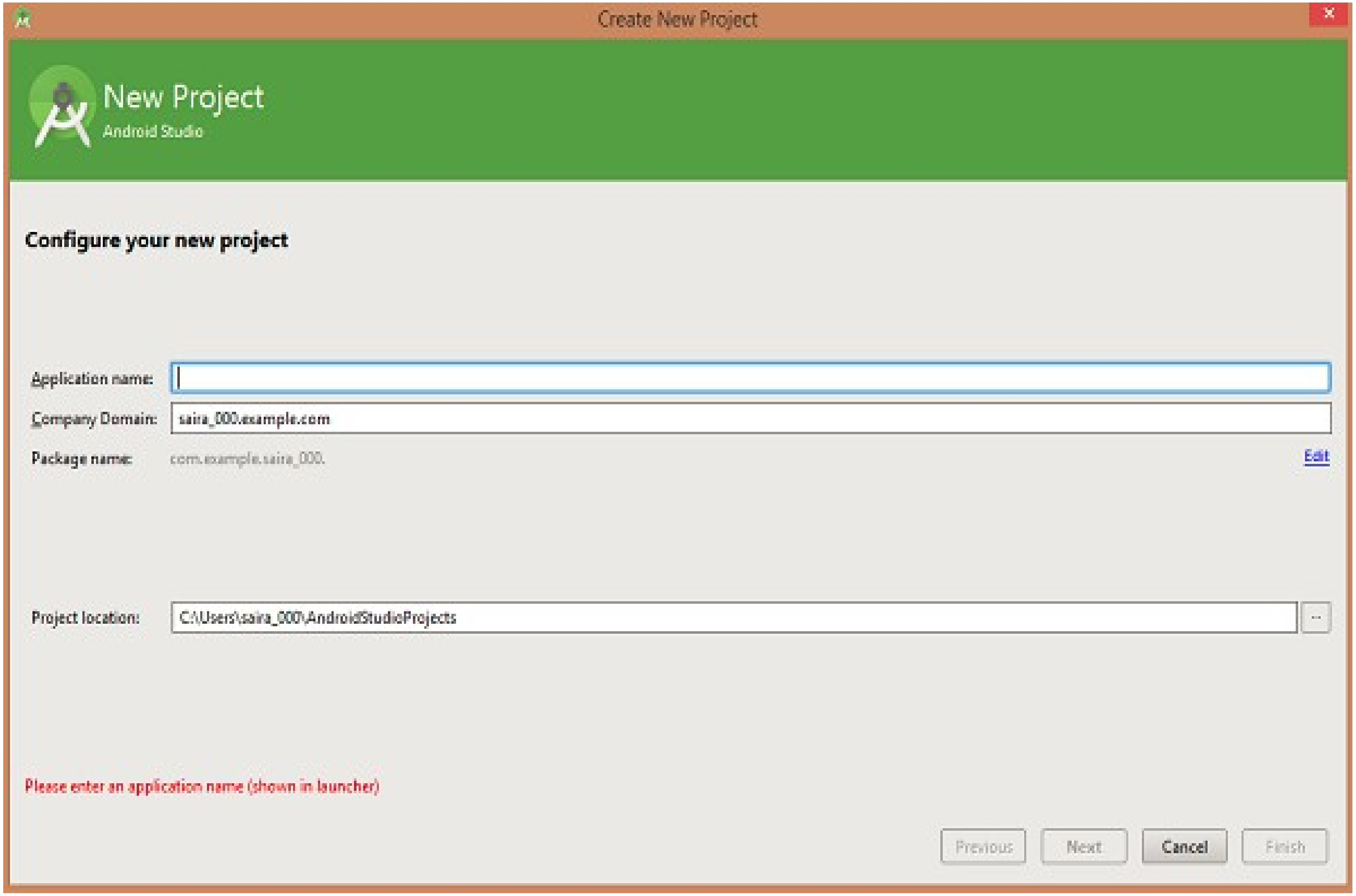
Application components are the essential building blocks of an Android application. These components are loosely coupled by the application manifest file *AndroidManifest.xml* that describes each component of the application and how they interact.

Components	Description
Activities	They dictate the UI and handle the user interaction to the smartphone screen
Services	They handle background processing associated with an application.
Broadcast Receivers	They handle communication between Android OS and applications.
Content Providers	They handle data and database management issues.

Components	**Description**
Fragments	Represent a behavior or a portion of user interface in an Activity.
Views	UI elements that are drawn onscreen including buttons, lists forms etc.
Layouts	View hierarchies that control screen format and appearance of the views.
Intents	Messages wiring components together.
Resources	External elements, such as strings, constants and drawable pictures.
Manifest	Configuration file for the application.

Hello World Program in Android


Android Studio Setup Wizard
Welcome to Android Studio
Recent Projects
No Project Open Yet
Quick Start
Start a new Android Studio project
Open an existing Android Studio project
Import an Android code sample
VCS
Check out project from Version Control
Import project (Eclipse ADT, Gradle, etc.)
Configure
Docs and How-Tos



Configure your new project

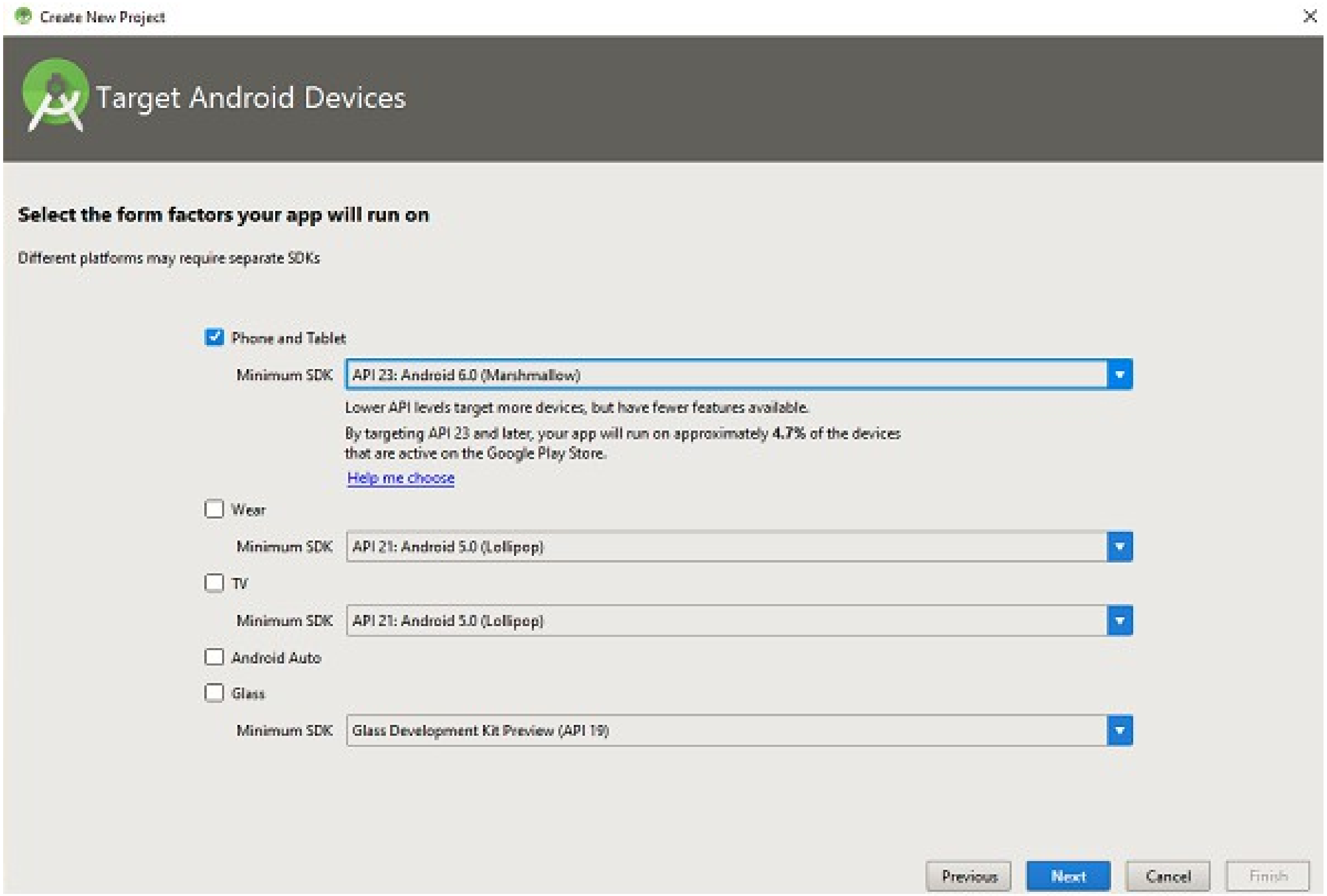
Application name:

Company Domain: saira_000.example.com

Package name: com.example.saira_000.

Project location: C:\Users\saira_000\AndroidStudioProjects

Please enter an application name (shown in launcher)



Create New Project

Target Android Devices

Select the form factors your app will run on

Different platforms may require separate SDKs

[✓] Phone and Tablet

Minimum SDK: API 23: Android 6.0 (Marshmallow)

Lower API levels target more devices, but have fewer features available.

By targeting API 23 and later, your app will run on approximately **4.7%** of the devices that are active on the Google Play Store.

Help me choose

[ ] Wear

Minimum SDK: API 21: Android 5.0 (Lollipop)

[ ] TV

Minimum SDK: API 21: Android 5.0 (Lollipop)

[ ] Android Auto

[ ] Glass

Minimum SDK: Glass Development Kit Preview (API 19)

Project Structure

- AndroidManifest.xml

  This is the manifest file which describes the fundamental characteristics of the app and defines each of its components.

- Java

  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.

Project Structure

- res/drawable

  This is a directory for drawable objects that are designed for high-density screens.
- res/layout

  This is a directory for files that define your app's user interface.
- res/values

  This is a directory for other various XML files that contain a collection of resources, such as strings and colours definitions.

Project Structure

- Build.gradle

  This is an auto generated file which contains compileSdkVersion, buildToolsVersion, applicationId, minSdkVersion, targetSdkVersion, versionCode and versionName

Code Structure

- MainActivity.java

```
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);
```

- R.layout.activity_main refers to the activity_main.xml file located in the res/layout folder

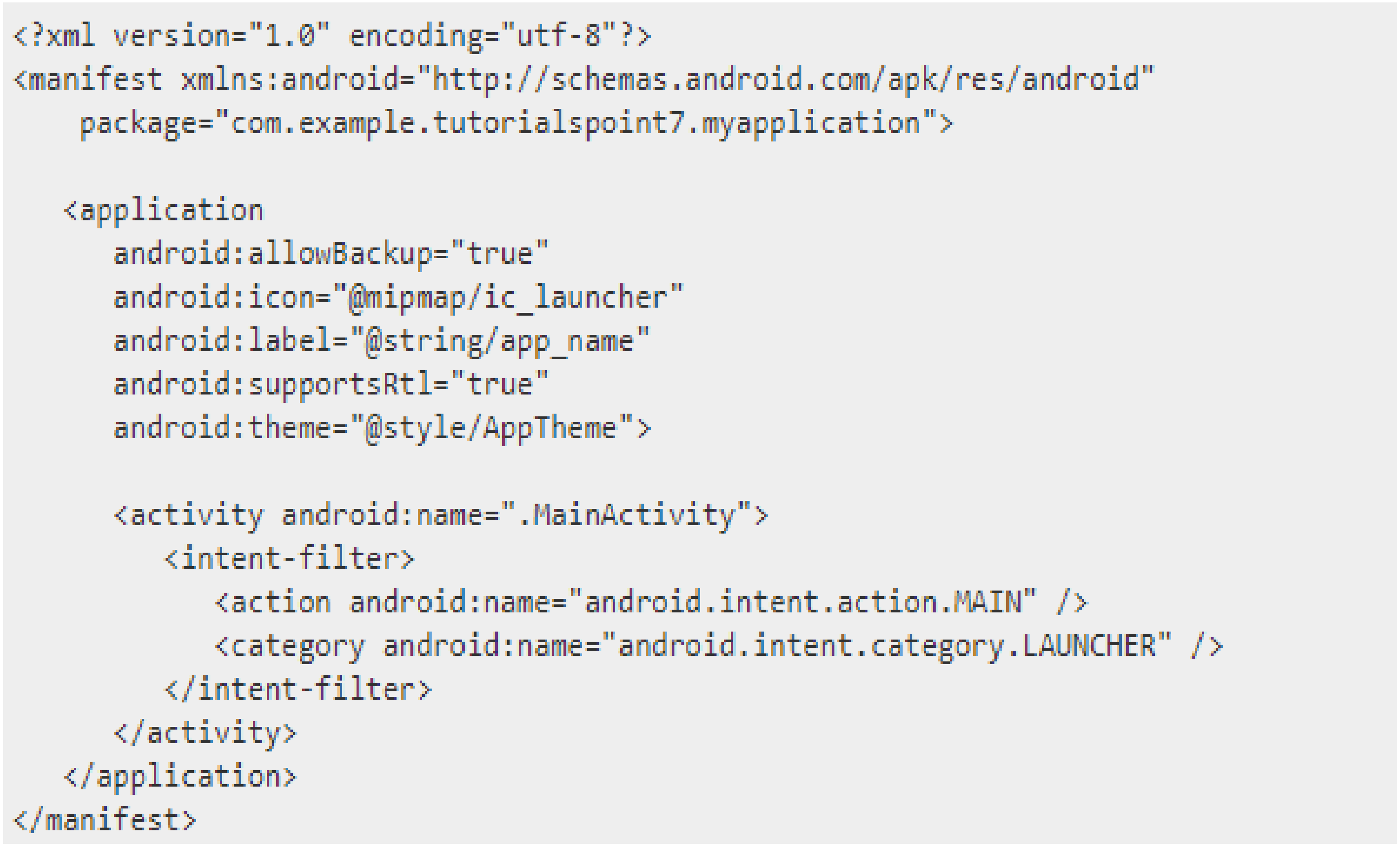
Code Structure

- Manifest File

  -- You must declare all its components in a manifest.xml which resides at the root of the application project directory.

  -- This file works as an interface between Android OS and your application, so if you do not declare your component in this file, then it will not be considered by the OS.

Code Structure : Manifest file



```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.tutorialspoint7.myapplication">

   <application
      android:allowBackup="true"
      android:icon="@mipmap/ic_launcher"
      android:label="@string/app_name"
      android:supportsRtl="true"
      android:theme="@style/AppTheme">

      <activity android:name=".MainActivity">
         <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
         </intent-filter>
      </activity>
   </application>
</manifest>
```

Code Structure

- Manifest File

  -- Here <application>...</application> tags enclosed the components related to the application.

  -- android:icon

  -- <activity> tag

Intent-filter conetents

- The action for the intent filter is named android.intent.action.MAIN -- indicate that this activity serves as the entry point for the application.
- The category for the intent-filter is named android.intent.category.LAUNCHER to indicate that the application can be launched from the device's launcher icon.

String Reference

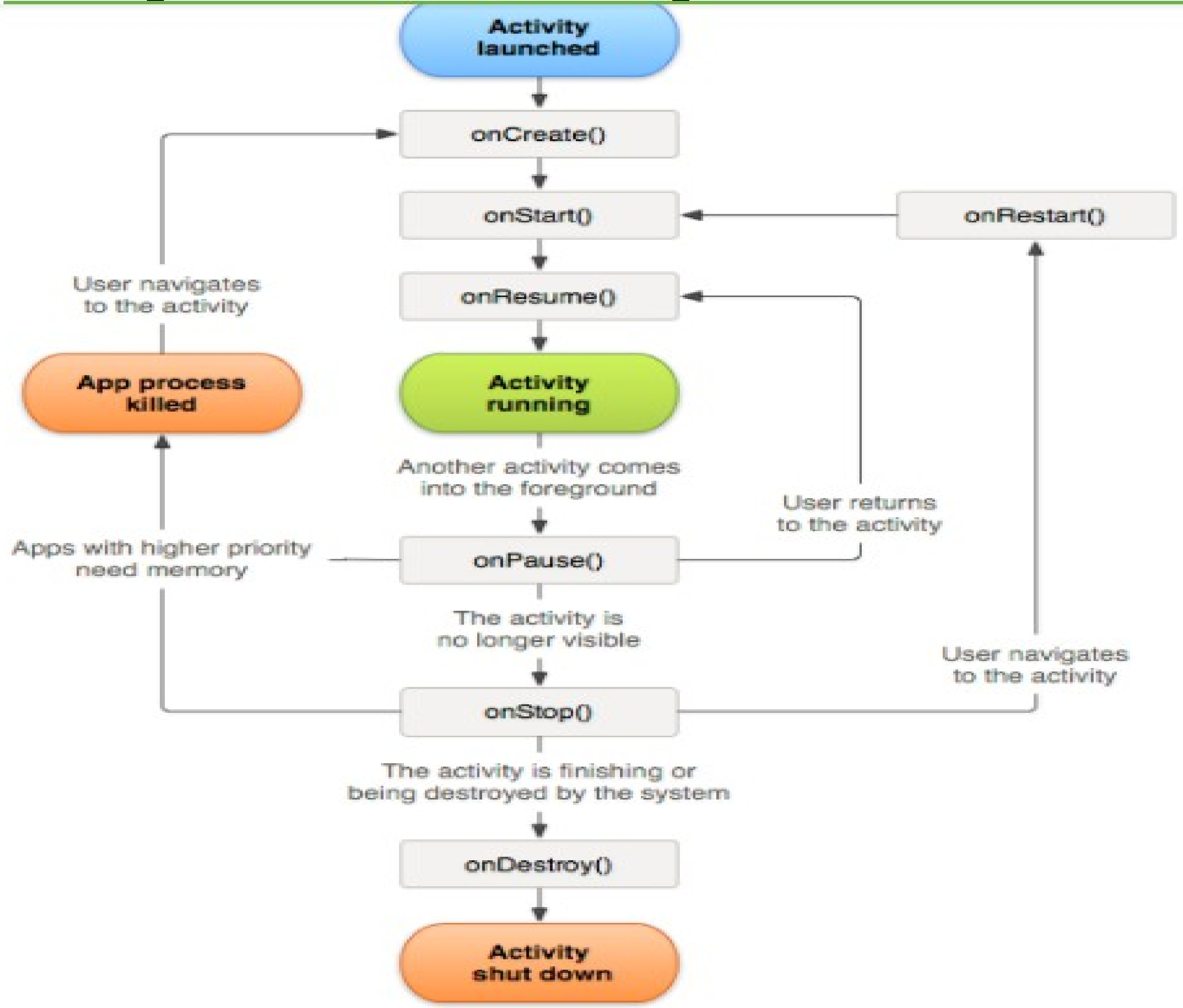
- The @string refers to the strings.xml file explained below. Hence, @string/app_name refers to the app_name string defined in the strings.xml file
- Strings.xml file

```
<resources>

<string name="app_name">Welcome</string>

</resources>
```

Different Android application components

- <activity>elements for activities
- <service> elements for services
- <receiver> elements for broadcast receivers
- <provider> elements for content providers

Life Cycle of Activity


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