**1. Gradle Files in an Android Project**

There are **two Gradle files** in an Android project:

MyAndroidApp/

│── build.gradle (Project-Level)

│── app/

│ ├── build.gradle (Module-Level)

│── gradle.properties

│── settings.gradle

│── gradlew / gradlew.bat

**a) build.gradle (Project-Level)**

Located in the **root directory**, it manages **global configurations** like:

* Gradle version
* Plugin dependencies
* Repository sources

**Example**

// Root-level build.gradle

buildscript {

dependencies {

classpath 'com.android.tools.build:gradle:8.1.0' // Android Gradle Plugin

}

}

allprojects {

repositories {

google() // Google's Maven Repository

mavenCentral() // Central repository for dependencies

}

}

**b) build.gradle (Module-Level)**

Located inside the app/ module, it configures:

* **SDK versions**
* **Dependencies**
* **Build types (debug/release)**
* **Product flavors**

**Example**

// Module-level build.gradle

plugins {

id 'com.android.application'

id 'kotlin-android' // If using Kotlin

}

android {

compileSdk 34 // Target Android SDK version

defaultConfig {

applicationId "com.example.myapp"

minSdk 21 // Minimum Android version supported

targetSdk 34 // Recommended SDK version

versionCode 1

versionName "1.0"

testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"

}

buildTypes {

debug {

applicationIdSuffix ".debug" // Different ID for debug builds

debuggable true // Allows debugging

}

release {

minifyEnabled true // Enables code shrinking

proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'

}

}

}

dependencies {

implementation 'androidx.core:core-ktx:1.10.1'

implementation 'androidx.appcompat:appcompat:1.6.1'

implementation 'com.google.android.material:material:1.9.0'

}

**c) settings.gradle**

Defines the project structure and includes modules.

// settings.gradle

rootProject.name = "MyAndroidApp"

include ':app'

**d) gradle.properties**

Used for **performance optimizations and custom properties**.

org.gradle.jvmargs=-Xmx2g # Increases max heap size

android.useAndroidX=true # Enables AndroidX

**e) gradlew and gradlew.bat**

These are **Gradle Wrapper scripts** that allow building the project without installing Gradle manually.

* gradlew → for **Linux/macOS**
* gradlew.bat → for **Windows**

✅ To build the app from the terminal:

./gradlew assembleDebug

✅ To clean the project:

./gradlew clean

**2. Gradle Dependencies**

Gradle manages external **libraries and dependencies** using:

dependencies {

implementation 'androidx.appcompat:appcompat:1.6.1'

implementation 'com.google.code.gson:gson:2.10'

}

**Types of Dependencies**

* implementation → Compiles the dependency into the app.
* api → Same as implementation but exposes it to dependent modules.
* compileOnly → Used only at compile time.
* testImplementation → Includes dependencies for **unit testing**.
* androidTestImplementation → Includes dependencies for **UI testing**.

**3. Build Types (Debug vs Release)**

Gradle supports **different build configurations**.

buildTypes {

debug {

debuggable true

}

release {

minifyEnabled true // Shrinks and obfuscates the code

proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'

}

}

**4. Product Flavors (Multiple App Variants)**

Product flavors allow creating **different app versions (e.g., free vs pro)**.

flavorDimensions "version"

productFlavors {

free {

applicationIdSuffix ".free"

versionNameSuffix "-free"

}

pro {

applicationIdSuffix ".pro"

versionNameSuffix "-pro"

}

}

Now, you can build **free** and **pro** versions separately:

./gradlew assembleFree

./gradlew assemblePro

**5. ProGuard (Code Shrinking & Obfuscation)**

In release builds, **ProGuard** helps **reduce APK size** and **protect the code**.

release {

minifyEnabled true

proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'

}

ProGuard rules are defined in:

app/proguard-rules.pro

Example rule:

-keep class com.example.myapp.\*\* { \*; }

**6. Gradle Tasks**

You can run various Gradle tasks:

| **Command** | **Description** |
| --- | --- |
| ./gradlew build | Compiles and builds the app |
| ./gradlew clean | Removes build artifacts |
| ./gradlew assembleDebug | Creates a debug APK |
| ./gradlew assembleRelease | Creates a release APK |
| ./gradlew dependencies | Lists project dependencies |

**7. Dependency Management**

You can **update dependencies automatically** using:

./gradlew dependencies

or specify a dependency in versions.gradle:

ext {

retrofitVersion = '2.9.0'

}

dependencies {

implementation "com.squareup.retrofit2:retrofit:$retrofitVersion"

}

**8. Custom Gradle Tasks**

You can define **custom tasks** to automate processes.

task printHello {

doLast {

println 'Hello, Android Developer!'

}

}

Run it with:

./gradlew printHello

**Conclusion**

* **Gradle automates the build process** for Android apps.
* **Project-Level build.gradle** defines global settings.
* **Module-Level build.gradle** handles dependencies, SDK versions, and build types.
* **Gradle Wrapper (gradlew)** ensures compatibility across machines.
* **Product flavors** enable multiple app versions.
* **ProGuard** optimizes APK size and security.