Gradle Notes for Beginners

1. Introduction to Gradle

What is Gradle?

Gradle is an **open-source build automation tool** that helps developers automate the process of building, testing, and deploying applications. It is designed to handle projects of any size, from small scripts to large enterprise-level applications.

Why do we need Gradle?

In software development, building an application involves many steps: - Compiling the source code - Managing project dependencies (libraries and tools your app needs) - Running tests - Packaging the app (e.g., into a JAR, WAR, or APK file) - Deploying to servers or app stores

Doing all of this manually is time-consuming and error-prone. **Gradle automates these steps**.

Key Features of Gradle

1. **Flexibility** – Unlike older build tools (Ant, Maven), Gradle allows you to customize tasks easily.
2. **Dependency Management** – Handles downloading and managing libraries from repositories (like Maven Central or Google's repository).
3. **Multi-Project Support** – Perfect for large projects with multiple modules.
4. **Incremental Builds** – Only rebuilds parts of the project that changed, making builds faster.
5. **Plugin System** – You can extend Gradle with plugins (e.g., Java plugin, Android plugin).

Gradle vs. Other Build Tools

- **Ant** – Very flexible but requires a lot of manual configuration.
- **Maven** – Provides structure and dependency management but is less flexible.
- **Gradle** – Combines the best of both: structured, yet highly customizable.

Gradle's Build Language

- Gradle uses **Groovy** or **Kotlin DSL (Domain Specific Language)** for writing build scripts.
- Build scripts are usually found in a file named `build.gradle` (for Groovy) or `build.gradle.kts` (for Kotlin).

Example (`build.gradle` in Groovy):

```
plugins {
    id 'java'
}

group 'com.example'
version '1.0.0'

repositories {
    mavenCentral()
}

dependencies {
    implementation 'org.apache.commons:commons-lang3:3.12.0'
    testImplementation 'junit:junit:4.13.2'
}
```

How Gradle Works (Build Lifecycle)

1. **Initialization Phase** – Gradle figures out which projects are part of the build.
2. **Configuration Phase** – It reads all the build scripts and sets up tasks.
3. **Execution Phase** – It runs the tasks you asked for (e.g., compile, test, build).

Official Documentation Reference

Gradle's official documentation: https://docs.gradle.org

It provides: - **User Manual** – Explains concepts in detail. - **DSL Reference** – Full reference for available tasks and configurations. - **Samples** – Example projects to learn from.

✅ **In short:** Gradle is a powerful build tool that automates compiling, testing, and packaging applications. It is flexible, supports large projects, and is widely used (especially in Android development).