Build Systems: Maven and Gradle

Introduction to Build Systems

- - Automate compilation, testing, and deployment
- - Manage dependencies efficiently
- - Examples: Maven, Gradle, Ant
- - Essential for software projects

What is Maven?

- - A project management and build automation tool
- - Uses XML-based configuration (POM.xml)
- - Convention over Configuration
- - Centralized dependency management

Maven Project Structure

- - src/main/java → Source code
- - src/main/resources → Configuration files
- - src/test/java → Test cases
- - target/ → Compiled output

Maven Lifecycle

- 1. **Clean**: Removes previous builds
- 2. **Validate**: Check project structure
- 3. **Compile**: Converts source code to bytecode
- 4. **Test**: Runs unit tests
- 5. **Package**: Creates a JAR/WAR file
- 6. **Install**: Installs in local repository
- 7. **Deploy**: Deploys to remote repository

What is Gradle?

- - A flexible build automation tool
- - Uses Groovy or Kotlin-based DSL
- - Faster than Maven (incremental builds)
- - Used in Android development and enterprise applications

Gradle Project Structure

- - build.gradle → Build script
- - settings.gradle → Project settings
- - src/main/java → Source code
- - src/test/java → Test cases
- - build/ → Compiled output

Gradle Lifecycle

- - **Initialization**: Identifies project
- - **Configuration**: Evaluates build scripts
- - **Execution**: Runs tasks (compile, test, package)

Maven vs Gradle

Feature	**Maven**	**Gradle**
Language	XML (POM)	Groovy/Kotlin
Performance	Slower	Faster
Flexibility	Less flexible	Highly flexible
Popularity	Older, widely used	Gaining popularity
Android Support	No	Yes

Hands-on Example: Maven

- 1. Install Maven
- 2. Create a project using `mvn archetype:generate`
- 3. Add dependencies in `pom.xml`
- 4. Build using `mvn package`
- 5. Run tests using `mvn test`

Hands-on Example: Gradle

- 1. Install Gradle
- 2. Create a project using `gradle init`
- 3. Define dependencies in `build.gradle`
- 4. Build using `gradle build`
- 5. Run tests using `gradle test`

Best Practices

- - Use dependency management wisely
- - Keep build scripts clean and modular
- - Prefer Gradle for Android projects
- - Use Maven for enterprise-level Java applications
- - Automate testing and CI/CD integration