Lab Practical: Build a Simple Application using Gradle

Aim

This lab aims to introduce you to Gradle, a powerful build automation tool for Java projects. By
building a simple application, you'll gain hands-on experience with Gradle's core functionalities like
project creation, dependency management, and task execution.

Procedure

1. Setting Up the Environment

• Ensure you have Java (version 8 or above) installed on your system. You can verify this by
running `java -version` in your terminal.

• Download and install an IDE that supports Gradle plugins, such as IntelliJ IDEA or Eclipse.

2. Creating a Gradle Project

• Open your terminal and navigate to your desired project directory.

• Run the following command to initialize a new Gradle project:

```
gradle init --type java-application
```

This command will prompt you for some configuration options. Choose the following:

• **Project type:** Application

• **Implementation language:** Java

• **Source compatibility:** Choose a compatible Java version based on your system (e.g.,
Java 17)

Gradle will generate the essential project structure with a `build.gradle` file containing the build
configuration.

3. Writing the Application Code (Simple Greeter)

• Inside the `src/main/java` directory, create a package named `com.example.myapp` (or your
preferred package name).

• Within the package, create a new Java class named `Greeter.java`.

• Add the following code to `Greeter.java`:

Java

```java
package com.example.myapp;
```

```
public class Greeter {
public static void main(String[] args) {
System.out.println("Hello, World!");
}
}
```
This simple code defines a Greeter class with a main method that prints "Hello, World!" to the
console.

4. Configuring Dependencies (Optional)

• Gradle allows you to manage external libraries your application depends on. For instance, if
you wanted to use a logging library, you could add it to the build.gradle file:

```
Gradle
dependencies {
implementation 'org.slf4j:slf4j-api:1.8.0-beta4'
runtimeOnly 'org.slf4j:slf4j-simple:1.8.0-beta4'
}
```
This snippet defines two dependencies:

- slf4j-api: The core logging API
- slf4j-simple: A simple logging implementation

5. Building and Running the Application

• Open a terminal in your project directory.
• Execute the following Gradle task to compile your Java code:

```
gradle build
```
This task will compile the source code and create an executable JAR file (usually located in
build/libs).

• Run your application using the generated JAR:

```
java -jar build/libs/your-application-name.jar
```
Replace your-application-name.jar with the actual name of your JAR file.

6. Exploring Gradle Tasks

Gradle offers various built-in tasks that automate different stages of the development process.
Here are some commonly used tasks:

• clean: Deletes generated files like compiled classes and JARs.
• assemble: Builds your application, including compilation and packaging.

• run: Executes your application's main method.
You can explore these tasks by running gradle tasks in your terminal. The output will list all
available tasks and their descriptions.

7. Additional Considerations

• Gradle allows writing custom build scripts to automate complex tasks. Explore the Gradle
documentation for details on advanced build customization.
• Gradle integrates seamlessly with version control systems like Git, allowing you to manage
build configurations alongside your source code.

Inference

By completing this lab, you've gained practical experience with Gradle's core functionalities:
• **Project Creation:** Gradle simplifies project setup with pre-defined configurations for
common project types like Java applications.
• **Dependency Management:** Gradle handles external libraries your application relies on,
ensuring consistent versions and resolving conflicts.
• **Task Automation:** Gradle provides built-in and custom tasks to automate repetitive tasks
like compiling, building, and testing your application.
This lab has laid the groundwork for you to explore Gradle's vast capabilities in managing complex
build processes and integrating with continuous integration and delivery (CI/CD) pipelines for
efficient software development.

Result

This lab has successfully demonstrated how to build a simple Java application using Gradle.
You've created a project structure, written application code, configured dependencies (optional),
built and run the application, and explored some essential Gradle tasks.