**Experiment no.** 7

**Title**: Demonstrate Maven Build Life Cycle

**Objective:** To learn Maven Build Life Cycle.

**Theory :**

Maven is a widely-used build automation and project management tool in the Java ecosystem. It provides a clear and standardised build lifecycle for Java projects, allowing developers to perform various tasks such as compiling code, running tests, packaging applications, and deploying artefacts. This experiment aims to demonstrate the Maven build lifecycle and its different phases.

## Key Maven Concepts:

* Project Object Model (POM): The POM is an XML file named pom.xml that defines a project's configuration, dependencies, plugins, and goals. It serves as the project's blueprint and is at the core of Maven's functionality.
* Build Lifecycle: Maven follows a predefined sequence of phases and goals organized into build lifecycles. These lifecycles include clean, validate, compile, test, package, install, and deploy, among others.
* Plugin: Plugins are extensions that provide specific functionality to Maven. They enable tasks like compiling code, running tests, packaging artifacts, and deploying applications.
* Dependency Management: Maven simplifies dependency management by allowing developers to declare project dependencies in the POM file. Maven downloads these dependencies from repositories like Maven Central.
* Repository: A repository is a collection of artifacts (compiled libraries, JARs, etc.) that Maven uses to manage dependencies. Maven Central is a popular public repository, and organisations often maintain private repositories.

## Maven Build Life Cycle:

The Maven build process is organised into a set of build lifecycles, each comprising a sequence of phases. Here are the key build lifecycles and their associated phases:

## Clean Lifecycle:

* clean: Deletes the target directory, removing all build artifacts.

## Default Lifecycle:

* validate: Validates the project's structure.
* compile: Compiles the project's source code.
* test: Runs tests using a suitable testing framework.
* package: Packages the compiled code into a distributable format (e.g., JAR, WAR).
* verify: Runs checks on the package to verify its correctness.
* install: Installs the package to the local repository.
* deploy: Copies the final package to a remote repository for sharing.

## Site Lifecycle:

* site: Generates project documentation.

Each phase within a lifecycle is executed in sequence, and the build progresses from one phase to the next. Developers can customise build behaviour by configuring plugins and goals in the POM file.

## Materials:

* A computer with Maven installed (https://maven.apache.org/download.cgi)
* A code editor (e.g., Visual Studio Code, IntelliJ IDEA)
* Java Development Kit (JDK) installed (https://[www.oracle.com/java/technologies/javase-downloads.html)](http://www.oracle.com/java/technologies/javase-downloads.html))

## Experiment Steps:

**Step 1: Setup Maven and Java**

* Ensure that you have Maven and JDK installed on your system. You can verify their installations by running the following commands:

**mvn -v**

**java -version**

**Step 2: Create a Maven Java Project**

* Create a new directory for your project, e.g., MavenDemo.
* Inside the project directory, create a simple Java class, e.g., HelloWorld.java, with a main method that prints "Hello, Maven!".

***public class HelloWorld {***

***public static void main(String[] args) { System.out.println("Hello, Maven!");***

***}***

***}***

Create a pom.xml file (Maven Project Object Model) in the project directory. This file defines project metadata, dependencies, and build configurations. Here's a minimal example:

***<project xmlns="***[***http://maven.apache.org/POM/4.0.0***](http://maven.apache.org/POM/4.0.0)***" xmlns:xsi="***[***http://www.w3.org/2001/XMLSchema-instance***](http://www.w3.org/2001/XMLSchema-instance)***"***

***xsi:schemaLocation="***[***http://maven.apache.org/POM/4.0.0***](http://maven.apache.org/POM/4.0.0)[***http://maven.apache.org/xsd/maven-4.0.0.xsd***](http://maven.apache.org/xsd/maven-4.0.0.xsd)***">***

***<modelVersion>4.0.0</modelVersion>***

***<groupId>com.example</groupId>***

***<artifactId>MavenDemo</artifactId>***

***<version>1.0-SNAPSHOT</version>***

***</project>***

## Step 3: Explore the Maven Build Phases

* Maven has several build phases, and each phase is responsible for specific tasks. In this step, we'll explore some of the most commonly used build phases.
* Clean Phase: To clean the project (remove generated files), run:

***mvn clean***

* Compile Phase: To compile the Java source code, run:

## mvn compile

* Test Phase: To execute unit tests, run:

***mvn test***

* Package Phase: To package the application into a JAR file, run:

***mvn package***

* Install Phase: To install the project artifacts (e.g., JAR) into your local Maven repository, run:

***mvn install***

* Deploy Phase: To deploy artifacts to a remote Maven repository, configure your pom.xml and run:

***mvn deploy***

## Step 4: Run the Application

* After running the mvn package command, you can find the generated JAR file (e.g., MavenDemo-1.0-SNAPSHOT.jar) in the target directory. Run the application