MAVEN ASSIGNMENT

1.Done under the project folder simplejavaproject.
2.Done under the project folder maven_first.
3.Done under parent project using modules.
4.

Plugins- Plugins are the central feature of Maven that allow for the reuse of common build logic across multiple projects. They do this by executing an "action" (i.e. creating a WAR file or compiling unit tests) in the context of a project's description - the Project Object Model (POM). Plugin behavior can be customized through a set of unique parameters which are exposed by a description of each plugin goal (or Mojo).

dependencies- the packages we import from different repositories, or from our own local machine, or other developer's code.

parent- parent tag defines the parent for the current module in it's POM

profile- essential for differentiating the usage and behaviour of the modules in a project under different environments.

properties- Maven properties are value placeholder, like properties in Ant. Their values are accessible anywhere within a POM by using the notation ${X}, where X is the property.

modules- define hierarchy in the project.

groupId- will identify your project uniquely among all the projects. Simple java package naming conventions should be used while giving groupId to a project.

artifactId- A project’s artifactId is often used as the name of a deliverable. It is the name of the JAR file without version.

modelVersion- It's defined as a mandatory, possibly to enforce a specific XML model in case new models are defined.
A POM has to comply with a model.

packaging- The most important aspect of a Maven project is its packaging type, which specifies the type of artifact the project produces. There are many built-in Maven packaging types (for example, jar, war, and ear).
A project's packaging type specifies the plugin goals that are executed during each Maven build phase.
For example, in a jar project, the maven-jar-plugin's jar goal is executed. In a war project, the maven-war-plugin's war goal is executed.

version- This tells us the version of the project. Snapshot represents that the project is in development. When deployed, Snapshot is removed from "version". Version can change with major as well as minor changes.

5.use the projects parent to see inheritance. Projects parent and child together show the concept of Aggregation.

6.Comparision of Gradle and Maven.
a)Gradle uses domain-specific language based on the programming language Groovy, differentiating it from Apache Maven, which uses XML for its project configuration.
b)Gradle is based on a graph of task dependencies – in which tasks are the things that do the work – while Maven is based on a fixed and linear model of phases.
c)Performance-wise, both allow for multi-module builds to run in parallel. However, Gradle allows for incremental builds because it checks which tasks are updated or not. If it is, then the task is not executed, giving you a much shorter build time.
d)Gradle uses a compiler daemon that makes compiling a lot faster.

7.Use of mvn clean install command:
`Mvn clean install` tells Maven to do the `clean` phase in each module before running the `install` phase for each module.