DAY 29:

ASS 1: Build Lifecycle

Demonstrate the use of Maven lifecycle phases (clean, compile, test, package, install, deploy) by executing them on a sample project and documenting what happens in each phase.

ANSWER:

Below is a simplified example of a Java project structure that demonstrates the use of Maven lifecycle phases. For simplicity, let's assume we have a single class `Main.java` with a simple `main` method.

Project Structure:

```
project/
 ├── src/
|   └── main/
|       └── java/
|           └── Main.java
└── pom.xml
```

Here's what each phase does:

1. clean: This phase cleans up the project directory by deleting the `target` directory which contains the compiled classes and generated artifacts.

2. compile: This phase compiles the source code and generates the `.class` files. It checks for any errors in the source code.

3. test: This phase runs the tests found in the `src/test` directory. If any test fails, it stops the build process.

4. package: This phase packages the compiled code (`.class` files) into a distributable format, such as JAR, WAR, or EAR. It creates the artifact in the `target` directory.

5. install: This phase installs the packaged artifact into the local Maven repository, making it available for other projects that are built on the same machine.

6. deploy: This phase copies the final package to the remote repository for sharing with other developers or projects.

Let's execute these phases on our sample project:

1. clean:

bash

```
mvn clean
```

   - This command deletes the `target` directory if it exists.

2. compile:

bash

```
mvn compile
```

   - Maven compiles the source code (`Main.java`) and generates the `.class` file(s) in the `target/classes` directory.

3. test:

bash

```
mvn test
```

   - Maven runs any tests in the `src/test` directory. Since we don't have any tests in this example, it will skip this phase.

4. package:

bash

```
mvn package
```

   - Maven packages the compiled code into a JAR file (or another format if specified) and places it in the `target` directory.

5. install:

bash

mvn install

   - Maven installs the packaged JAR into the local Maven repository (`~/.m2/repository`) so that other projects can use it as a dependency.

6. deploy:

bash

mvn deploy

   - This phase typically involves deploying the artifact to a remote repository, such as Nexus or Artifactory. Since we're not connected to a remote repository in this example, it won't do anything meaningful.

That's how Maven lifecycle phases work on a sample Java project!