Unit Test and JaCoCo

Add code below into the end of file pom.xml

```
<plugin>
        <groupId>org.jacoco</groupId>
        <artifactId>jacoco-maven-plugin</artifactId>
        <version>0.8.5</version>
        <executions>
          <execution>
            <goals>
              <goal>prepare-agent</goal>
            </goals>
          </execution>
          <execution>
            <id>report</id>
            <phase>test</phase>
            <goals>
              <goal>report</goal>
            </goals>
          </execution>
        </executions>
</plugin>
```

This code is part of a Maven plugin called Jacoco, which is a code coverage tool used to measure Java code. This particular snippet configures the executions of the plugin. It runs the "prepare-agent" goal first, and the "report" goal after the test phase is complete. The "prepare-agent" ensures that Jacoco is running properly, while the "report" creates a report at the end of the test phase which contains coverage details. This helps to determine how well the code is tested.

```
stage('Unit Tests') {
        steps {
          sh "mvn test"
        }
        post {
          always {
            junit 'target/surefire-reports/*.xml'
            jacoco execPattern: 'target/jacoco.exec'
          }
        }
    }
```

This code is part of a Jenkins pipeline that runs unit tests on a Java project. The "stage" block is used to define a set of actions that should be performed in order, in this case, running the unit tests and generating a report. The "sh" command runs the "mvn test" command which triggers the actual unit tests. The "post" block is used to publish the results of those tests. The "junit" command will publish the results of the unit tests to Jenkins in the form of XML files. The "jacoco" command is used to generate a code coverage report, based on the "execPattern" specified. This code coverage reports will allow developers to measure how well their code is executing.
The **`always`** block in the Jenkins stage code you provided is an element of the Pipeline Syntax. It is used to define the activities that should be executed regardless of the outcome of the stage (i.e. success or failure). In your example code, this block includes the steps to copy the Junit test reports and generate the Jacoco code coverage file.
The `jacoco execPattern: 'target/jacoco.exec'` code is used to generate a code coverage report

for a Java project. Jacoco uses an execution pattern to monitor the Java code under test and create an executable report. 'Target/jacoco.exec' represents the path to the Jacoco execution input file. This file contains information collected by Jacoco to measure the code coverage achieved by the tests. The report tells developers which segments of the code were executed and which were not. It also provides details on lines covered, missed, as as percentage of coverage achieved. It is an important tool to help developers measure and improve the quality of their code.