Objectives

- Unit testing with Junit5
- Maven (adding dependencies to pom.xml)
- Code coverage with JaCoCo

Maven

- In this assignment, your code need to be compiled using only Maven in a command line. You can use Eclipse, but your code will be graded using Maven. Your pom.xml must be modified to include extra dependencies.
- Maven can also generate coverage reports using JaCoCo plugin. We have already added this plugin for you in pom.xml (take a look into your provided pom.xml and understand how the plugin is added to the build).
- To run your tests with JaCoCo and produce code coverage results, you can go to where your pom.xml is and execute the following commands:
    - mvn test
    - mvn jacoco:report
- The first command runs your unit tests, and the second command generates human readable reports of coverage information of your unit tests. The reports will be stored in the "target/site/jacoco" directory.
- You may need to install Maven to run it in a command line. We refer to https://maven.apache.org/ for download instructions and tutorials.

Overview

- The goal of this assignment is to create a library management application with unit tests, evaluate quality of these tests, and automate build.
- You will only create a simple model for this library application
    - Books are organized in collections.
    - A book can be added to or removed from a collection.
    - A collection contains a various number of books.
    - A collection can also contain other sub-collections.

- For example, the Computer Science collection can contain a series of books about CS in general and other sub-collections such as Operating Systems, Programming Languages, Software Engineering, etc.

Skeleton code

- Download the attached homework2.zip. After unzipping the files, you should see a pom.xml file and an src folder.
- The src folder containing the skeleton for the source code and test files. There are 4 classes in the source code(Book.java, Collection.java, Element.java, Library.java). You should implement all methods with TODO.
- **DO NOT** modify the API of any provided methods (e.g., do not change the signature of public methods). You can add more methods if you want.
- We strongly recommend that you use some library for XML, JSON, YML, etc., instead of writing your own code to parse strings.
- You will **need to modify pom.xml** to include extra dependencies you need (some library for reading/writing structured strings).
- **Please read the description in the source code very carefully.**

Tests

- You should implement the code for the tests provided in BookTest.java, CollectionTest.java and LibraryTest.java.
- Optionally, you can add more tests. The naming of extra tests should follow the existing tests' naming convention.
- We will run students' code and tests against each other (as appropriate, based on the library dependencies), and we will have some teaching staff code and tests to run against your code and tests.
- You **MUST** ensure that your tests pass on your code; you will get overall 0 points if your submitted code and tests do not work with "mvn test".
  - Please see the junit 5 user guide when you write the test cases. (https://junit.org/junit5/docs/current/user-guide/)

Coverage

- Your submitted tests need to achieve at least **80%** statement coverage on every class in the directory "src/main" of your submitted code.
- Use JaCoCo to measure the statement coverage of your code with the tests. If you do not have enough coverage, add more tests.
- When you have achieved more than 80% coverage, upload the JaCoCo report in CSV format from "target/site/jacoco/jacoco.csv".

Turning in

- Create a private project with name "homework2" in the GitLab repository http://gladius.postech.ac.kr, and clone the project on your machine.
- Commit your changes in your "homework2" project and push them to the remote repository that includes your pom.xml file and your src folder.
  - homework2/jacoco.csv generated by "mvn jacoco:report"
- You have to tag the project with "submitted" when you finish your homework before the deadline (see https://docs.gitlab.com/ee/university/training/topics/tags.html).