Assignment01-Q3: There are many beneficial features of Maven that are used for Java. Specifically, it has improved performance for Java-based projects, making project building and managing much faster and easier for developers. In addition, it incorporates management simplicity for Java projects by automating downloads as well as dependency management. This ensures that the libraries or JAR files are the correct version and well maintained through repositories. Moreover, Maven enables developers to build and publish various projects simultaneously with its efficient handling of modules. It also integrates very well with IDEs as the IDEs automatically recognize the Maven pom.xml file and configures the project by building paths and project structures without the need for manual, human intervention. Thus, it is extensible and makes it easy for the developer to write plug-ins. Conclusively, Maven is a very useful tool and provides many benefits for Java projects.

Assignment01-Q4: The “Divide And Conquer Principle” refers to breaking down a complex problem into smaller, manageable sub-problems. This enables developers to focus on the solutions of such sub-problems individually, making it easier to test and maintain the code. After that, developers can integrate the individual solutions to each sub-problem to solve the complex problem. Another benefit of this principle is reusability, in which developers can reuse independent classes multiple times in an efficient manner. Ultimately, this results in the code being simple and easier to comprehend as it is organized into smaller parts.

The “Encapsulation Principle” ensures that an object’s internal state is modified only through well-defined methods. This improves its security as the internal details of the object are further hidden through access control, more specifically, access modifiers (such as private and public). Thus, this principle allows developers to modify the inside of classes without affecting other parts of the program that rely on the object's behaviour.

The “Interface Principle” defines contracts for classes that adhere to the interface contract. The principle uses methods to create loosely coupled, interchangeable components, which means it allows for easy substitution of implementations without affecting dependent code. Essentially, it reduces the risk of breaking code when changes are made and parts of the system interact with the interface.

The “Open-Closed Principle” refers to the notion that existing code should be closed for modification, meaning new functionality can be added to a class without actually changing the existing code. In addition, the principle refers that the existing code should be open for extension, where a class or function can be extended with new behaviour. This makes it easy to test or debug because adding onto debugged existing code will only result in a smaller part that needs debugging and testing. Therefore, the principle enhances code stability and reduces the risk of introducing new bugs.