INF221 Project – Phase II
Architecture-Based Development with Java 9

A. Case Study:

In the first phase of this project, you migrated a Java application to Java Platform Module System (provided by Java 9+), through which the Java application's components and their dependencies are explicitly specified in terms of Java modules and their module-info files. Java modules are intended to make it easier for developers to improve the encapsulation, security, and maintainability of large Java applications. Software designers and developers can achieve strong encapsulation and high maintainability by modularizing Java applications which allows them to explicitly specify which of a Java module's public, protected, and private are accessible or inaccessible by other modules. It reduces the complexity of the system's architecture, enhances its encapsulation and increases its maintainability.
JPMS has added more refined accessibility control—allowing architects and developers to decrease accessibility to packages, reduce the points at which a Java application may be susceptible to security attacks, and design more elegant and logical architectures.

In the second phase of this project, you need to take the Java application you chose for the first phase and provide the scenario of a possible issue, bug, or vulnerability within the application before migration which is fixed by modularizing the application and specifying its dependencies and accessibilities.

The scenario could be of a maintenance issue regarding the encapsulation of the modules, for instance, where a possible change within the system that propagates across the components introduces some errors or faults. Alternatively, the scenario could explain a security issue caused by the internal APIs of a system being accessed or modified by other components. The detailed description of the second phase of this project is as follows.

- Provide the scenario of a possible issue or bug that may arise within the Java application and explain:
    1. The proposed issue, bug, or vulnerability
    2. The concrete and detailed scenario in which the proposed issue occurs
    3. What has changed within the Java application regarding the proposed scenario after migration to Java 9+
    4. How the change, mentioned in 3, resolve the issue, specified in 1

Tip: Explore the issues on the application's GitHub repositories to find the scenario of an existing bug or issue that is fixed after your migration.

B. Architectural Inconsistencies

In the process of migrating a Java application to Java 9+, the descriptive architecture of the implemented system may be inconsistent with the prescriptive architecture provided in its module-info files. Such inconsistencies may arise due to a misunderstanding of a software systems' architectures (e.g. mistakenly specifying a more accessible interface than intended) or intentionally for possible uses of the application. A report of some possible architectural inconsistencies within the Java application, which you migrated in the previous section, is shared with you through Canvas. Provide short answers for the following questions regarding each inconsistency detected in your Java application. (If there were no inconsistencies identified in your application, you may skip this part of the project.)

1. Is the identified inconsistency correct?
2. Why is the excess dependency specified in the module-info file?

Note that each group has to present their project, both phase I and II on Dec 5th, and 15% of the grade for phase II is the presentation of your work.