**LOUIE JAY A. CENIZA MARCH 22, 2024**

**EMERGENCE (CHAPTER 12)**

**Getting Clean via Emergent Design.** It talks about the four rules of Kent Beck, that is significant of helping to create a well-designed software.

**Simple Design Rule 1: Runs All the Tests.** Emphasizes the importance of ensuring that a system is testable in order to verify its functionality. A system that cannot be effectively tested should not be deployed. By making systems testable, developers are pushed towards creating smaller, single-purpose classes that adhere to the Single Responsibility Principle (SRP), leading to better designs.

**Simple Design Rules 2–4: Refactoring.** Once tests are in place, the process of maintaining clean code involves continuous refactoring. After adding a few lines of code, developers pause to assess the impact on the design and consider whether it has been degraded. If necessary, they refactor the code, cleaning it up while ensuring that tests still pass, thus eliminating the fear of breaking anything. During refactoring, various principles and techniques of good software design are applied, such as increasing cohesion, decreasing coupling, separating concerns, modularizing system concerns, reducing function and class sizes, selecting clearer names, and more.

**No Duplication.** Avoid duplication as it leads to additional work, risks, and unnecessary complexity. Duplication can take various forms, such as identical lines of code, similar code that can be further refined, or duplicated implementation. By identifying and removing duplication, code becomes cleaner and more maintainable.

**Expressive.** As the name implies you need a word that can describe your naming conventions so that it can be understand easily and read easily.

**Minimal Classes and Methods.** Keep your classes and methods small and simple. However, even though your classes and methods are small you need to ensure that the individual functions and classes are concise and focused.