# Week 3

## SOLID (lecture hour 1)

1. What are the five SOLID principles?
    i. Single Responsibility Principle
    ii. Open/Closed Principle
    iii. Liskov Substitution Principle
    iv. Interface Segregation Principle
    v. Dependency Inversion Principle
2. For each of the following examples from the lecture slides, write the SOLID principle that it exemplifies:
    - Employee class = Single Responsibility Principle
    - Area Calculator = Open/Closed Principle
    - Rectangle and Square classes = Liskov Substitution Principle
    - Manager, Worker, and IWorker = Dependency Inversion Principle
3. Which of the five SOLID principles are you most confused about and why? (opinion question)
   The Interface Segregation Principle, because no example was provided for the principle. I am guessing that it is saying we should not put all methods in a same class/interface if some objects of the class / that extend the interface don't really need them.
4. Which of the five SOLID principles do you feel is the most important? (opinion question)
   The Dependency Inversion Principle. No lower-level objects should call a method from the higher-level class, or things would be really messy.

## Clean Architecture (lecture hour 2)

5. What are the four layers of Clean Architecture?
    i. Enterprise Business Rules (Entities)
    ii. Application Business Rules (Use Cases)
    iii. Interface Adapters (Controllers, Gateways, Presenters)
    iv. Frameworks & Drivers (UI, DB, Devices, Web, External Interfaces)
6. For each of the following classes from CleanArchitectureDemo, state in which layer of Clean Architecture they belong:
    - SystemInOut = Gateway
    - Student = Entities
    - Course = Entities
    - StudentManager = Use Cases
    - EnrolmentSystem = Controllers
    - StudentPropertiesIterator = Presenter
7. What is the Dependency Rule?

The outer layers depend on the inner layers. The dependence within the same layer is allowed, but we should try to minimize coupling. The name of something (functions, classes, variables) declared in an outer layer must not be mentioned by the code in an inner layer.

8. What is one way to identify a violation of the Dependency Rule in your code?
    - Look at the imports at the top of each source file in your project.
    - For example, if you see that you are importing a Controller class from inside an Entity class, that is a violation of clean architecture! Likewise, if you see a Controller referencing an Entity that is also likely a violation
9. Why is it advantageous to keep "details" in the outer layer and the entities in the inner most layer?
    - The business rules can be easily tested without worrying about those details in the outer layers!
    - Any changes in the outer layers don’t affect the business rules

## Java (course notes, quiz, github-4)

We encourage you to make any notes about Java concepts you found tricky. Please ask on piazza or during office hours if you have questions about any of the Java content from this week or the previous weeks.