Week 5 Notes

+ Exceptions (continued from last week)

1. If class A has a method (call it method1) that throws an Exception and class B has a method (call it method2) that calls method1, where does it make the most sense to put a try and catch block?
In method 2.

+ SOLID principles and Clean Architecture (Design)

2. What is a design principle? How is it different from a rule? (from last week)
Principles are ideals that can be worked towards, but you don't necessarily need to fulfill them, while rules are plainly ideas that are either followed or broken.

3. In your own words, explain each of:

- Single Responsibility Principle
  Every class should have only one reason to change (a single responsibility). Else there may be conflict with changes by different people.

- Open/Closed Principle
  The code should be written in a way that can be extended in terms of features but should not have to change the code that is already written.

- Liskov Substitution Principle
  If more subclasses of a parent classes are made, existing code with the parent should still work with objects of the child class.

- Interface Segregation Principle
  Classes should not have to implement irrelavent methods because of interfaces.

- Dependency Inversion Principle
  Parent classes shouldn't depend on child classes (ie shouldn't have to change because of new or changing child classes)
  The goal is to have low coupling in a program.

4. What do we mean when we say that we are implementing "Clean Architecture"?
Answer this question by explaining the meaning of each term:

- Entity: Base building blocks of the program, the very basic elements of a program.

- Use Case Class: Classes that store entities, and can possibly change values within them.

- Controller: Takes input from the user of the program and can use the Use Case Classes depending on user input.

- Presenter: Sends information to display to the user.

- Gateway: Sends and reads information from a database.

- UI: The visual display that users can see and interact with. (User Interface)

- Devices: Devices/Hardware that interact with the program.

- Database: Stores and sends information to the program.

- Web: Web pages that may recieve or send information to the program.

- External Interfaces: Other programs that interact with the program.

5. What is the Dependency Rule, when implementing Clean Architecture?
Dependencies can only point inwards. Web/Devices/DB/External Interfaces/UI can depend on

Gateways/Controllers/Presenters which can depend on Use Case Classes which can depend on Entities.

+ Intro to CRC

6. What is CRC?

Class, responsibility, collaboration

A tool/method for designing a program involving the discussion of the responsibilities and relationships of required classes.

7. How do you and your team use CRC to design a program together?

You can have meetings with your whole team to form ideas of how your program will be built before starting to code it.

8. How does the TicketVendor example CRC from lecture break the dependency Rule in Clean Architecture?

ShowManager is dependent on Show which is dependent on Ticket Manager which is also dependent on Ticket. So a use case class is dependent on an entity which is dependent on a use case class.

9. Do you acknowledge that Phase 0 of the project is to be submitted individually? You can discuss it with the other people on your team, but your words should be your own and not copied off of someone else.

I acknowledged.

10. List at least three possible Entity class names for the TicketVendor program. Can any of them be replaced by Strings?

Ticket

Performance

Seat

Seat could be replaced by the position of the seat, as ticket and seat have a lot of the same information.

11. What does it mean to compose a Controller class inside a Presenter class? What does it mean to compose a Presenter class inside a Controller class? Is this allowed in Clean Architecture?

This means there is a Controller class as a variable in a Presenter class and vice versa. This is allowed because Controllers and Presenters are in the same category in Clean Architecture.