Lecture_11

Single Responsibility Principle

- Classes should have one and only one job
- If this principle is broken, it becomes increasingly difficult to:
  - Track a bug or feature
  - Minimize the impact of requirement changes
  - Write automated unit tests
- A class should only have one reason to change

Liskov Substitution Principle

- The idea behind Liskov substitution is that a parent class can be replaced by a **subclass** without breaking the application
- Minimizes coupling and maximizes reuse

Interface Segregation Principle

- The more poorly implemented methods, the more likely that domain boundaries are not set correctly in your design.
- Major coding patterns will not function appropriately if interfaces are not implemented correctly
- If you consider what external users will see when trying to use your class it makes a lot of sense - if there are 200 methods, and 198 of them have empty implementations, there's a design problem
- General printer interface should be replaced with **Print, Scan, Fax**

Dependency Inversion Principle: Why?

- A Patient monitor class is tightly coupled with specific alert mechanisms
- Implement the dependency inversion principle by introducing an alert interface, serving as an abstraction to decouple patient monitor from specific alert mechanisms, enhancing flexibility.