Design Defects and Restructuring

LECTURE 06

SAT, OCT 10, 2020

Design Principles – SOLID

Single Responsibility Principle

Open Close Principle

Liskov Substitution Principle

Interface Segregation Principle

Dependency Inversion Principle

The Single Responsibility Principle

A class should have one reason to change

Responsibility: a reason for change

Why? Because each responsibility is an axis of change

When the requirements change, that change will be manifest through a change in responsibility amongst the classes

If a class assumes more than one responsibility, then there will be more than one reason for it to change

If a class has more than one responsibility, then responsibilities become coupled

This kind of coupling lead to fragile design that break in unexpected ways when changed

The Open – Close Principle

Software entities (classes, functions, modules, etc.) should be open for extension, but closed for modification

Open for extension: This means that the behavior of the module can be extended

Closed for modification: Extending the behavior of a module does not result in changes to the source or binary code of the module

The Liskov Substitution Principle

Subtypes must be substitutable for their base types

A violation of LSP is a latent violation of OCP

The Interface Segregation Principle

Clients should not be forced to depend upon methods which they do not use

Interfaces belong to clients, not to hierarchies

The Dependency Inversion Principle

High level modules should not depend upon low level modules

- Both should depend upon abstraction

Abstractions should not depend upon details

- Details should depend upon abstractions

Depend on Abstractions

- No variable should hold a pointer or reference to a concrete class
- No class should derive from a concrete class
- No method should override an implemented method of any of its base class