Low Level Design (LLD)

Part 5

S.O.L.I.D. Principles

These five software development principles are guidelines to follow when building software so that it is easier to scale and maintain. They were made popular by a software engineer, Robert C. Martin.

S — Single Responsibility

A class should have a single responsibility

O — Open-Closed

Classes should be open for extension, but closed for modification

L — Liskov Substitution

If S is a subtype of T, then objects of type T in a program may be replaced with objects of type S without altering any of the desirable properties of that program.

I — Interface Segregation

Clients should not be forced to depend on methods that they do not use.

D — Dependency Inversion

High-level modules should not depend on low-level modules. Both should depend on the abstraction.

Abstractions should not depend on details. Details should depend on abstractions.

Factory Pattern

Factory Pattern

The factory method pattern is a design pattern used in object-oriented programming to create objects without specifying the exact class of the object that will be created. It provides an interface for creating objects in a superclass but allows subclasses to alter the type of objects that will be created.

Factory Pattern

Create objects without specifying the exact class of the object that will be created.

Factory Pattern

Create objects without specifying the exact class of the object that will be created.

- The type of the object is determined at runtime.
- You need pointer or reference to a base class.

Factory Pattern

Create objects without specifying the exact class of the object that will be created.

- The type of the object is determined at runtime.
- You need pointer or reference to a base class.

POLYMORPHISM!

Factory Pattern

It provides an interface for creating objects in a superclass but allows subclasses to alter the type of objects that will be created.

Factory Pattern

When it is useful?

- When you have a class that can have multiple subclasses, each providing a different implementation.
- When you want to decouple the client code from the concrete classes being instantiated.
- When you need to extend the application by adding new types of objects without modifying existing code.

Example

Let’s go to VS Code!

Thank You!