Lecture_12

Design Patterns

Abstract classes are superclasses which contain abstract methods and are defined such that concrete subclasses extend them by implementing the methods.

In object oriented programming, an interface or protocol is a data type that acts as an abstraction of a class.

Factory pattern

- A creational design pattern that provides an interface for creating objects without specifying the exact class of the object that will be created
- The main goal is to decouple the object
- **EX**
  - We have different shapes, and we want to create instances of these shapes without exposing the construction logic directly to the client code

Abstract Factory Pattern

- More complex patter
- Provides an interface for creating families of related or dependent objects without specifying their concrete class
- Enhance flexibility, maintain loose coupling, consistency

Factory Pattern - Use When

- The creation of objects should be independent of the system utilizing them
- The systems should be capable of using multiple families of obj

---

Singleton Pattern

- Ensures a class only has one instance of itself.
- The constructor is a private member function

- **Ex**
  - Create a logger system that helps track application events and aids developers in identifying and diagnosing issues during runtime
  - There are three games created, they all share the same leaderboards. It doesn't matter how the games were created.

Lazy INIT

- Do not create an object until it is about to be used
- If an object is never used its never created
- Reduce risk of multiple initialization

Drawbacks

- Global state can introduce potential risks and potential issues with maintaining state consistency across different parts of the application
- In multithreaded environments, there is no guarantee that a method will run to completion before a method in another thread starts running

When to use

- Exactly one instance of a class is needed throughout the system
- Multiple instances would incur loading a massive amount of duplicate data into memory

---

Adapter Pattern

- Structural design pattern that allows two incompatible interfaces to work together by acting as a bridge between them, It is used when you have an existing class with different interface and want to use it in a context where another interface is expected.
- It is often used during transition periods
- Enables code reusability
- Can add to the complexity of a solution

Use When

- Unable to use one class or component directly
- Modifying one element would cause significant issues

Facade Pattern

- a software-design pattern commonly used in object-oriented programming. Analogous to a facade in architecture, a facade is an object that serves as a front-facing interface masking more complex underlying or structural code.
- Facade design pattern is used to help client applications to easily interact with the system

Use When

- A simple interface is needed to provide access to a complex system
- There are many dependencies between system implementations and clients
- Systems and subsystems should be layered

Open Closed Principle

software entities (classes, modules, functions, etc.) should be open for extension, but closed for modification_is, such an entity can allow its behavior our to be extended without modifying its source code.