Design pattern :-

Pros:-

* Independent on language
* Simplify common task
* Well studies for pros and cons
* It is specification not implementation

Category of design pattern :-

* Creational design pattern
* Structural design pattern
* Behavior design pattern
* Initation of Object

Creational Design pattern :-

Pros :-

1. Hide the object creational logic
2. Maintain object logic/life cycle

Types of Creational Design pattern :-

* Singleton
* Templet
* Factory
* Abstract factory
* Proxy
* Builder design pattern

Singleton design pattern:-

* Create the object with certain properties and sate which will we preserve through out the program life time.
* No other object is possible to create further.

Example :- java Date API

Implementation:-

* Double checking
* Read resolve 🡪 Avoid serialization
* Override clone method
* Best practice Enum 🡪 to avoid reflection to break singleton

Prototype Design pattern:-

* Create object with similar state and behavior
* Without using new keyword
* It could we clone or serializable

Example:-Spring use implement for bean creation

Factory design pattern:-

* Their should be a common interface and different implemented class
* Central factory will provide object of desire type

Example :- Ice cream

Abstract Factory:-

1. It is called factories of factory
2. First calling Factory will return factory object
3. Return factory object will return actual object

Example:-

🡪Amul 🡪 chocolate

Dairy 🡪Mother dairy -🡪Ice cream 🡪 strawberry

🡪India dairy 🡪 Nuts creams

Builder design pattern :-

Pros :-

* Creation of complex object using multi stage process/ hierarchal steps

Example :-

Creation of Home 🡪 Painting

* Design
* Decorating

Structure Design pattern :-

Structure design pattern solve the problem related to modifaction extension

And marger of software unit.

Common Structural design pattern are –

* Adapter
* Façade
* Proxy
* Bridge
* Composite

Adapter :-

This is use to connect two sub unit with incompatible type into a single unit with help of adapter.

Adapter could be an interface with provide common platfrom for client

Example:-

Google Phone 🡪 Adapter 🡪 Ear phone common

Iphone 🡪

Façade Design pattern:-

* This pattern help to solve the problem different services with common platform
* Façade pattern add an interface to existing system or group of subsystem to hide its complexity
* This pattern involves a single class which provide method with simplified method required by client and delegate call to method of existing system call
* Façade wapper over Existing implementation
* Microservices and RestAPI

Example :- lets consider we have different cloud system and their method and implementation, this make very complicated for client to manage and understating to the implementation, so to resolve the problem make a façade class which will take care for all the services and hide the services complexity and expose common method for all the services.

Proxy:- 🡺 Representing

Proxy 🡺 on behalf of

* In place of

Proxy is used when object creation with complete functionality is a deep graph then instead of creating original object create proxy object that will implement common interface as original interface.

Proxy is wapper or an agent that is called by client to access the real se

Example :- phone

* Demo phone
* Real phone

Iterator design pattern:-

Iterator pattern is used iterate the collection element in a sequential order in uniform manner.

Java collection API uses the iterator pattern over collection framework beautifully

Example:- lets assume we have a collection of student and client want all student details instead of sharing of collection object preferred way to share of Iterator of that collection object.

Composite Design pattern:-

* Composite design pattern is used when client need to ignore the difference between composition of object and individual object.
* Recursive composition
* This pattern create that contain group of own object
* Composite pattern compose object in term of tree structural to represent as well as whole hierarchy

Example:- Organization Structure

Bridge Design pattern:-

* Decouple to connect component
* Example:- TV ,Remote
* When minor modification in TV does not need to change remote and vice-versa
* Bridge pattern decouple are abstraction two can vary independently In short convert ‘Is-a’ relation into Has-a relation.

Mediator Design pattern:-

* This design pattern is used when, two or more unit are want to communicate and communication is complex.
* Example :- Flight takeover
* Mediator help to established communication or co-ordination

Decorator Design pattern:-

* This design pattern is used when add the new functionality with out altering the old structure
* Their will be a decorator class which is wapper for original class
* Its support open for extension closed for modification

Example:- Microservices over rest services for validation over user.