Design Patterns

A Solution to General Problems Facing Software Developers

What We Will Cover...

- Creational Patterns
  - Singleton
  - Builder
  - Factory
- Structural Patterns
  - Decorator
  - Adapter
  - Proxy
  - Facade
- Behavioral Patterns
  - Observer
  - Strategy
  - Template
  - Command

Singleton Pattern

Defines a class that only has one instance, and provides a global point of access.

This pattern saves memory because the object is not created at every request.

Used in multi-threaded and database applications, as well as logging, caching, and configuration settings.

Builder Pattern

Constructs a complex object from simple objects step by step.

Provides separation between the construction and representation of a given object.

Used when objects can't be created in a single step.

Factory Pattern

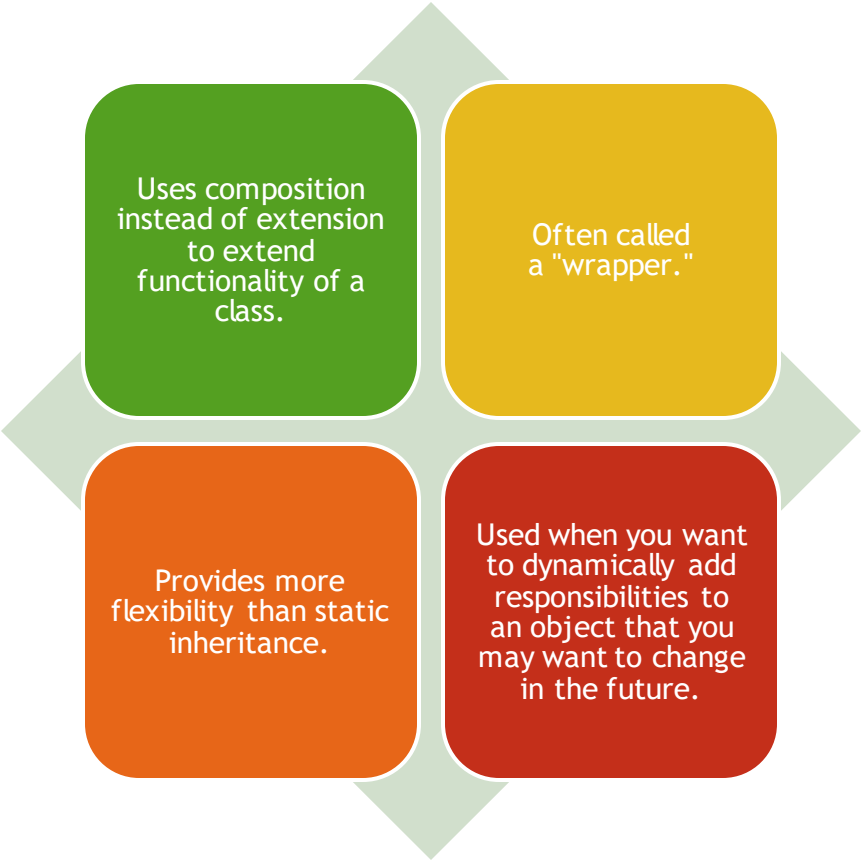
Defines an interface or abstract class for creating objects but allows the sub classes to decide which class to instantiate.

Allows sub classes to choose the type of objects to create.

Used when a class doesn't know what sub classes will be required to create or the class wants the sub classes to specify what objects to create.

Now, on to Structural Patterns…

Decorator Pattern



CONVERTS THE INTERFACE OF A CLASS SO IT CAN INTERACT WITH METHODS OF A CLASS WITH A DIFFERENT INTERFACE.

ALLOWS TWO OR MORE INCOMPATIBLE OBJECTS TO INTERACT WITH EACH OTHER.

ALSO CALLED A "WRAPPER."

Adapter Pattern

Proxy Pattern

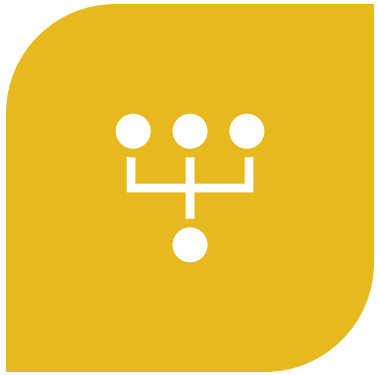
Hides the information and operations of the original objects by providing a placeholder object.

Provides protection of the original object from users.

Can be used in various senarios, including virtual proxy, protective proxy, and remote proxy.

Façade Patterns

A HIGH-LEVEL INTERFACE THAT HIDES THE COMPLEXITY OF THE SUB SYSTEM.



SHIELDS CLIENTS FROM COMPLEX SUB SYSTEMS IN A PROGRAM.

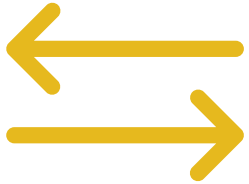
USED WHEN YOU WANT TO PROVIDE A SIMPLE INTERFACE WITHIN A COMPLEX SUB-SYSTEM.

Behavioral Patterns

Observer Pattern

Defines one to one dependancy between objects.

When one changes state, all others are updated.

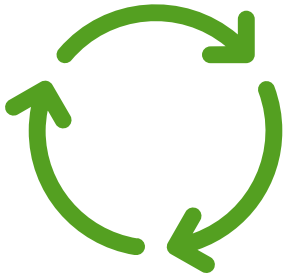


Used when the change of state in one object has to be reflected in another.

Strategy Pattern

- Changes the behavior of an algorithm at runtime.
- Can change the behavior of a class without extending it.
- Can be used to implement payment methods in an online store for example.

Template Pattern



Defines the skeleton of an algorithm but allows sub-classes override specific steps within it.

Used mainly to reuse code.

SEPERATES THE OBJECT THAT INVOKES AN OPERATION FROM THE ONE THAT ACTUALLY CARRIES IT OUT.

USED IN TRANSACTIONAL FUNCTIONALITY.

MAKES IT EASY TO ADD NEW COMMANDS WITHOUT ALTERING CLASSES.

Command Pattern