Most Used Design Patterns Cheat Sheet

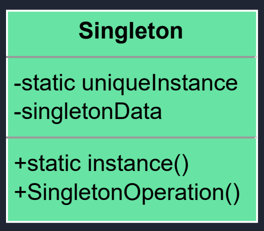
🟩 **Creational Patterns**
Used to construct objects

🟪 **Structural Patterns**
Used to form large object structures

🟨 **Behavioral Patterns**
Used to manage algorithms and relationships

Singleton

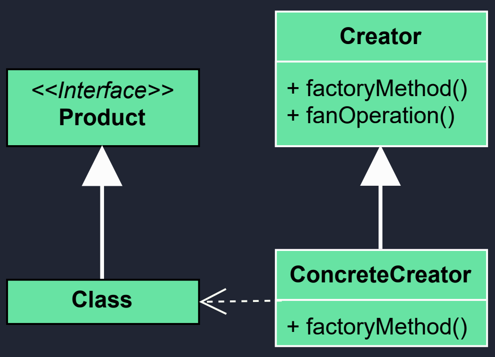
Use when you want to have one instances of a class.



Example: logging, db connections.

Factory Method

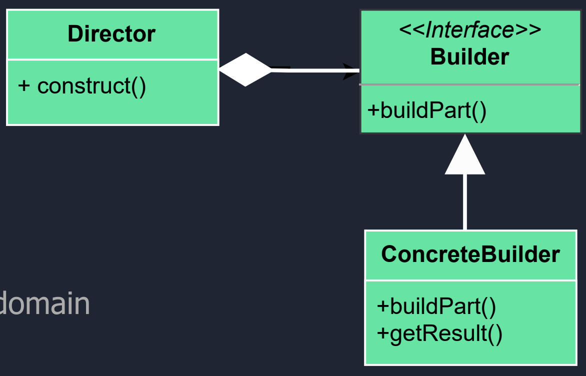
Use when you want to delegate object creation to subclasses.



Example: create GUI component

Builder

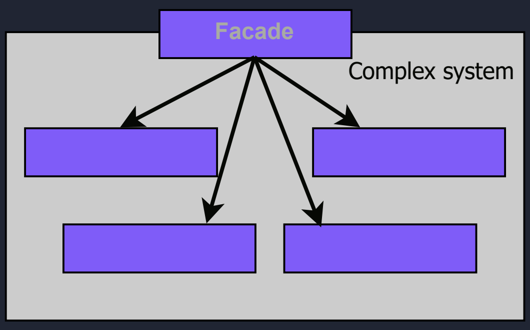
Constructing complex objects, step by step



Example: create complex domain object

Facade

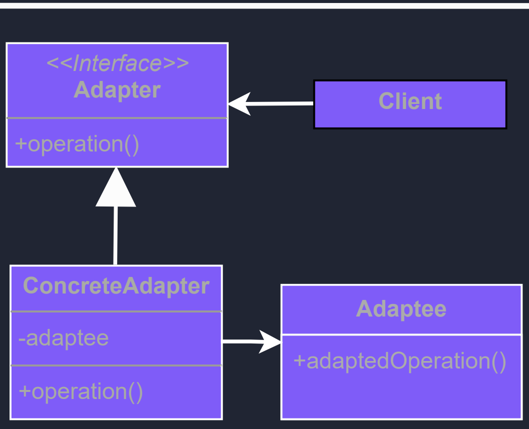
Use when you want to provide a simplified interface to a complex subsystem



Example: Providing a simple interface to a complex subsystem

Adapter

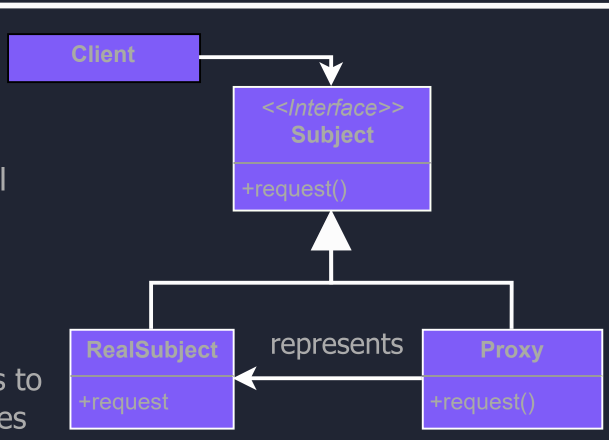
Use when you need to convert an interface to another interface



Example: make incompatible classes work toghether

Proxy

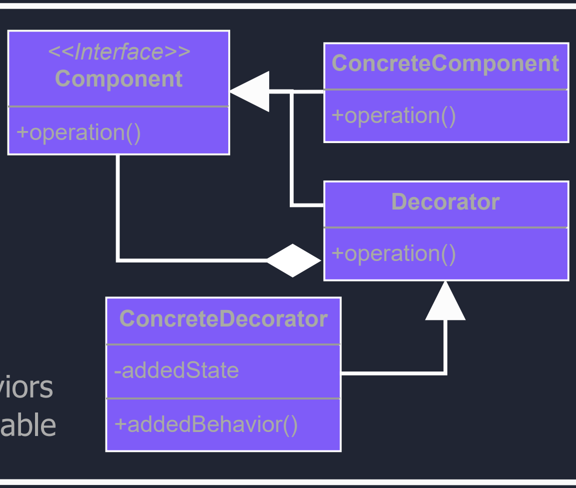
Use for object access control



Example: Controling access to sensitive resources

Decorator

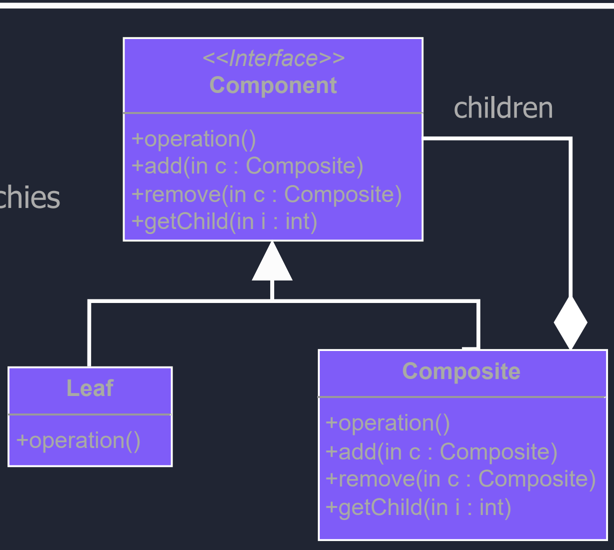
Use when you need to wrap objects to modify their behaviors



Example: make object behaviors dynamically modifiable

Composite

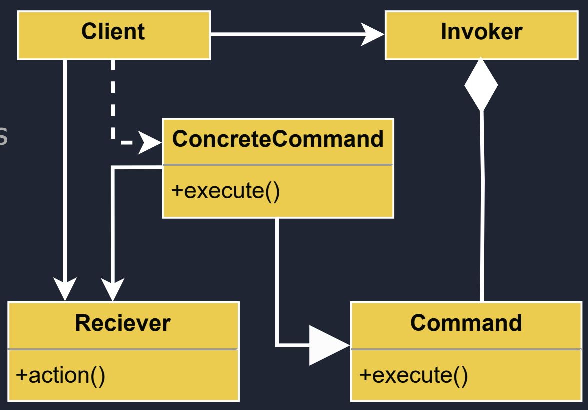
Represent part-whole hierarchies



Example: Graphic object in a drawing can be grouped

Command

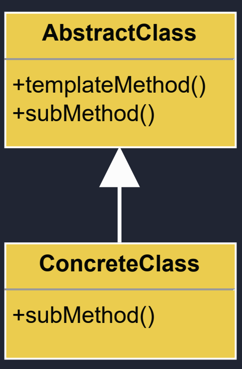
Use for encapsulating requests with parameters



Example: Implementing operations

Template Method

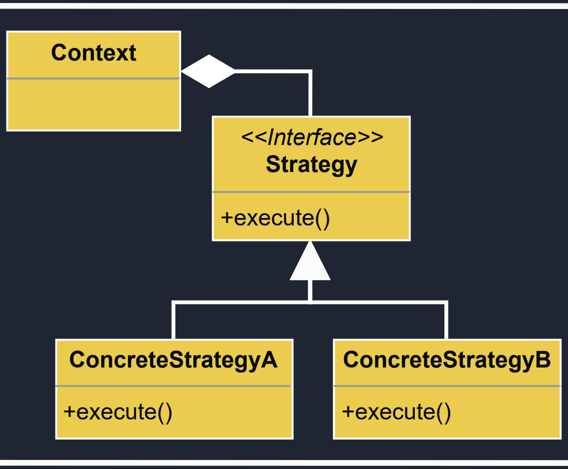
Use when you want to break down an algorithm into a series of steps



Example: Common behavior should be located in one class

Strategy

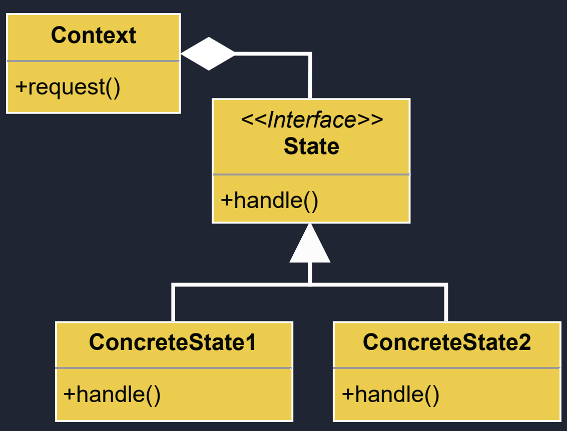
Use for interchangeable algorithms that can be swaped at runtime



Example: Implement different sorting algorithms

State

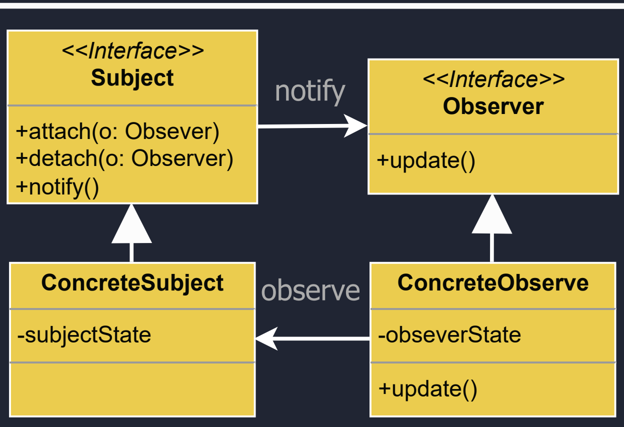
Encapsulate state-specific behavior



Example: Handling different states of a user interface

Observer

Use for automatic updates of dependand objects



Example: Implement subscribers

TechWorld WithMilan
simplifying complex topics