Design Patterns

Ethan Maiorini

Creational

- Singleton
- Builder
- Factory

Singelton

Description: A class that has one instance that gets used globally

Uses: When you need to use something multiple times but only need one instance of it to operate your program. Like a connection to the internet or database. Like if you needed multiple classes to track one object that they all used.

Builder

Description: Separates the construction from it’s representation

Uses: When you have one common item but many different but options in how its put together but a similar method of obtaining the parts. Like an auto factory, it builds cars but there are different options to each car but in the end still a car.

Factory

Description: One creator that allows the subclasses to do instantiation

Uses:When you need to create multiple objects but do not need to care about the details. An example would be like a bottling procedure dosent care what liquid goes in the bottle.

Structural

- Decorator
- Adapter
- Proxy
- Facade

Decorator

Description: Attaches additional responsibilities to an object

Uses: The decorator pattern is used to add functionality to a class much like putting a case on your phone to give it protection. Or when writing a program that needs extra functionality out of a class that you already defined, instead of writing a whole new class you could just make a decorator class that adds the functionality to your code and save yourself from re-writing code.

Adapter

Description: Converts the interface of a class into another

Uses: Taking an older or not exactly what the class needs interface and making it work for what you currently need lika a cable that goes from hdmi to rgb or vga. You use something you already have with something new you want to use.

Proxy

Description: Provides a surrogate placeholder for another object

Uses: Either makes sure that an object is protect and only accessed by the proxy. Or it is used to instantiate an object that is expensive to create and only gets made when needed.

Facade

Description: Provides a unified interface to a set of interfaces in a subsystem

Uses: Used to make a new Interface to bring a bunch classes together. Like an main system that attaches and directs you to the correct area.

Behavioral

- Observer
- Strategy
- Template
- Command

Observer

Description: Define a one to many dependence between objects where a state change in one object notifies all it dependencies

Uses: Used to notify other objects when states changed. Like when someone pings everyone on discord or just a smaller group. The people who need to know get the notification.

Strategy

Description: Define a family of algorithms that encapsulate each one and makes them interchangeable

Uses: To be able to interchange the algorithms based on the problem at hand to best suit the variables of the situation. An example might be a rideshare app that needs different size vehicles for different situations.

Template

Description: Defines the skeleton of an algorithm in an operation, deferring some steps to the subclasses

Uses: Ina situation where you want to layout the structure of where things go but leave the details at that point to the individual classes. Like a school schedule. You show up, go to lunch, and leave at the same points. Have classes at designated times but the classes themselves are left to the individual subjects to deal with the details.

Command

Description: Encapsulates a request as an object thereby allowing for the parameterization of clients with different requests

Uses: It creates a way in which you can give information or commands/orders to another class with an object to allow the request to be free of the knowledge needed to perform the task given.