23 Classic Gang of Four Design Patterns

Generated by ChatGPT:

Creational Patterns (deal with object creation)

1. **Singleton** - Ensures a class has only one instance and provides a global point of access to it.
2. **Factory Method** - Defines an interface for creating objects but lets subclasses decide which class to instantiate.
3. **Abstract Factory** - Provides an interface for creating families of related objects without specifying their concrete classes.
4. **Builder** - Separates the construction of a complex object from its representation, allowing the same construction process to create different representations.
5. **Prototype** - Creates new objects by copying an existing object (prototype) rather than creating from scratch.

Structural Patterns (deal with object composition)

6. **Adapter** - Converts the interface of one class into another interface that clients expect, enabling incompatible classes to work together.
7. **Bridge** - Decouples an abstraction from its implementation so they can vary independently, preventing a permanent binding between them.
8. **Composite** - Composes objects into tree structures to represent part-whole hierarchies, letting clients treat individual objects and compositions uniformly.
9. **Decorator** - Attaches additional responsibilities to an object dynamically, providing a flexible alternative to subclassing for extending functionality.
10. **Facade** - Provides a unified, simplified interface to a complex subsystem, making it easier to use.
11. **Flyweight** - Minimizes memory usage by sharing as much data as possible with similar objects; used when large numbers of objects are needed.
12. **Proxy** - Provides a surrogate or placeholder for another object to control access to it, useful for lazy initialization, access control, or logging.

Behavioral Patterns (deal with object interaction and responsibility)

13. **Chain of Responsibility** - Passes a request along a chain of handlers, where each handler decides either to process the request or pass it to the next handler.

14. **Command** - Encapsulates a request as an object, thereby allowing parameterization of clients with different requests and supporting undoable operations.
15. **Interpreter** - Defines a grammatical representation for a language and an interpreter to evaluate sentences in that language.
16. **Iterator** - Provides a way to access elements of an aggregate object sequentially without exposing its underlying representation.
17. **Mediator** - Defines an object that encapsulates how a set of objects interact, promoting loose coupling by preventing objects from referring to each other explicitly.
18. **Memento** - Captures and externalizes an object's internal state without violating encapsulation, so the object can be restored to this state later.
19. **Observer** - Defines a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically.
20. **State** - Allows an object to alter its behavior when its internal state changes, appearing to change its class.
21. **Strategy** - Defines a family of algorithms, encapsulates each one, and makes them interchangeable, letting the algorithm vary independently from clients.
22. **Template Method** - Defines the skeleton of an algorithm in a method, deferring some steps to subclasses, letting subclasses redefine certain steps without changing the algorithm's structure.
23. **Visitor** - Represents an operation to be performed on elements of an object structure, letting you define new operations without changing the classes of the elements.

These patterns provide proven solutions to common software design problems, promoting code reuse, flexibility, and maintainability in object-oriented systems.

Links to more external resources not mine...

> *I'm not affiliated with nor profit from these Books, they are just here for your and my convenience.*
>
> *Modern PHP 8 Design Patterns Online Version: https://designpatternsphp.readthedocs.io/en/latest/*
>
> *PDF Version: https://designpatternsphp.readthedocs.io/_/downloads/en/latest/pdf/*
>
> *Nice Book for sale: Buy Dive Into DESIGN PATTERNS An Ebook on design patterns and the principles behind them: https://refactoring.guru/design-patterns/book*
>
> *PHP Code from the Book above: https://refactoring.guru/design-patterns/php, If you liked it; Please support them by buying the book above...*
>
> *"The Gang of 4 book" - Design Patterns: Elements of Reusable Object-Oriented Software By*

> *Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, Published Oct 31, 1994. Buy here it: https://www.informit.com/store/design-patterns-elements-of-reusable-object-oriented-9780201633610*
>
> *More on The Addison-Wesley Professional Computing Series*