Search

What is Design Patterns?

Design patterns are reusable solutions to common problems in software desig n. They provide best practices and standardized approaches to solving recurri ng design challenges. Design patterns help improve(**code reusability, maintai nability, and scalability)** by offering tested and proven solutions.

Types of Design Patterns:

1_ **Creational Patterns** – Focus on object creation mechanisms.

- **Singleton**: Ensures only one instance of a class exists.
- **Factory Method**: Creates objects without specifying their exact class.
- **Abstract Factory**: Creates families of related objects.
- **Builder**: Constructs complex objects step by step.
- **Prototype**: Clones existing objects.

2_ **Structural Patterns** – Deal with object composition and relationships.

- **Adapter**: Allows incompatible interfaces to work together.
- **Bridge**: Separates abstraction from implementation.
- **Composite**: Treats individual objects and compositions uniformly.
- **Decorator**: Adds behavior dynamically to objects.
- **Facade**: Provides a simplified interface to a complex system.
- **Flyweight**: Reduces memory usage by sharing objects.
- **Proxy**: Controls access to an object.

3_ **Behavioral Patterns** – Focus on communication between objects.

- **Observer**: Notifies objects of state changes.
- **Strategy**: Selects an algorithm dynamically.
- **Command**: Encapsulates a request as an object.
- **Chain of Responsibility**: Passes requests through a chain of handlers.
- **Mediator**: Centralizes communication between objects.
- **Memento**: Saves and restores an object's state.
- **State**: Changes object behavior based on its state.
- **Template Method**: Defines a skeleton of an algorithm.
- **Visitor**: Allows adding operations to objects without modifying them.