Advanced PHP OOP Concepts

1. **Inheritance**
    - Allows one class to inherit properties and methods from another class.
    - Helps in reusing code and creating a parent-child relationship between classes.
2. **Abstract**
    - A class that cannot be instantiated directly.
    - Contains at least one abstract method (a method without implementation).
    - Child classes must implement the abstract methods.
3. **Static**
    - Properties and methods that belong to the class itself, not to an instance.
    - Can be accessed without creating an object using ClassName::method().
4. **Final**
    - Prevents a class from being extended or a method from being overridden.
    - Ensures that certain functionality remains unchanged in subclasses.
5. **Method Overloading**
    - Not directly supported in PHP like other languages.
    - Achieved using magic methods (__call or __callStatic) to handle dynamic method calls.

6. **Method Overriding**
    - When a child class provides a new version of a method inherited from a parent class.
    - The overridden method in the child class replaces the parent's implementation.
7. **Interface**
    - Defines a contract for classes by declaring methods without implementation.
    - A class that implements an interface must provide implementations for all its methods.
8. **Encapsulation**
    - Restricts direct access to class properties and methods.
    - Uses private, protected, and public visibility to control access.
9. **Polymorphism**
    - Allows different classes to be treated as instances of the same parent class.
    - Enables methods to work differently based on the object that calls them.

When to Use What?

- **Use an Abstract Class** when you have common functionality to share among multiple related classes.
- **Use an Interface** when you want to enforce a set of methods across multiple unrelated classes.

Why Use Static Classes, Properties, and Methods in PHP?

1. Static Class (A Class with Only Static Methods & Properties)

- Used when you don't need to create an object to access its methods.
- Often used for utility/helper classes (e.g., logging, mathematical operations, configuration handling).

2. Static Properties (Variables Belonging to the Class, Not Instances)

- Shared across all instances of the class.
- Useful for maintaining common values like counters, database connections, or configuration settings.

3. Static Methods (Functions That Belong to the Class, Not Instances)

- Can be called directly using ClassName::method(), without creating an object.
- Useful for utility functions (e.g., Math::sum(10, 20), Logger::write('error message')).

Purpose of Using Static in PHP

Memory Efficiency – No need to create multiple objects for simple tasks.
Global Access – Methods and properties can be accessed anywhere without instantiating the class.
Utility Functions – Perfect for helper functions like formatting, logging, and calculations.
Shared Data – When you want a property to be the same for all instances (e.g., a counter tracking object creation).