20 Practical OOP PHP Questions

# Class (Basic OOP)

1. Create a PHP class `Book` with title, author, and price properties, and a method to display book details.

2. Write a PHP program to set and get private properties of a `Laptop` class using getter and setter methods.

3. Create a class `BankAccount` with deposit and withdraw methods and display the balance.

4. Write a PHP program to demonstrate constructor and destructor in a `Mobile` class.

# Inheritance

1. Create a base class `Person` and a derived class `Student` that adds roll number and grade.

2. Write a PHP program to show method overriding using a `Vehicle` base class and a `Car` subclass.

3. Create a class `Employee` with salary property and extend it into a `Manager` class that calculates bonus.

4. Write a PHP program to demonstrate the use of parent constructor using `parent::\_\_construct()` in inheritance.

5. Make a class `Shape` and create subclasses `Circle` and `Rectangle` to calculate area (runtime polymorphism).

# Interface

1. Create an interface `Payment` with methods `pay()` and `refund()`. Implement it in `CreditCard` and `PayPal` classes.

2. Write a PHP program with an interface `Printable` that forces implementing classes (`Book`, `Magazine`) to define a `printDetails()` method.

3. Create an interface `Logger` with `logInfo()` and `logError()` methods and implement it in two classes: `FileLogger` and `DatabaseLogger`.

# Abstract Class

1. Write an abstract class `Animal` with an abstract method `sound()`. Implement it in `Dog` and `Cat` classes.

2. Create an abstract class `Shape` with an abstract method `area()` and implement it in `Triangle` and `Square`.

3. Make an abstract class `Employee` with an abstract method `calculateSalary()` and extend it for `FullTimeEmployee` and `PartTimeEmployee`.

# Static Members

1. Write a PHP program to create a `Counter` class with a static property to count how many objects are created.

2. Create a class `MathOperations` with static methods for `add()`, `subtract()`, and `multiply()`.

3. Write a PHP script to demonstrate static properties sharing values across multiple objects.

4. Make a `Configuration` class that stores settings in static properties and display them without creating an object.

5. Create a class `Database` with a static method `getConnection()` that simulates a database connection (singleton pattern idea).