Object Oriented Programming fundamentals

1. **Class and Object Creation**
    a. Defining classes and objects
    b. Constructors and properties

2. **Encapsulation**
    a. Private and public access modifiers
    b. Getters and setters

3. **Inheritance**
    a. Parent and child classes
    b. protected properties
    c. Overriding methods

4. **Polymorphism**
    a. Implementing interfaces
    b. Method overriding

5. **Abstract Classes**
    a. Defining and using abstract classes

b. Implementing abstract methods

6. **Static Properties & Methods**

a. Declaring and using static properties

b. Calling static methods

7. **Magic Methods**

a. __construct() for initialization

b. __toString() for string representation

8. **Dependency Injection**

a. Passing dependencies via constructors

b. Using composition for better code modularity

9. **Traits**

a. Defining and using traits

b. Code reuse with multiple classes

10. Exception Handling

- Custom exception classes
- Throwing and catching exceptions

Practice Problems per concept

Exercise 1: Class and Object Creation

Concepts Covered: Classes, Objects, Properties, Methods

- Create a class `Car` with properties: `$brand`, `$model`, and `$year`.
- Add a constructor to initialize these properties.
- Add a method `getCarInfo()` that returns a string with the car's details.
- Instantiate the `Car` class and display its details.

Exercise 2: Encapsulation

Concepts Covered: Private & Protected Properties, Getters, and Setters

- Modify the `Car` class to make all properties `private`.

- Create getter and setter methods for each property.

- Ensure that the setter for `$year` does not allow values earlier than 1886 (the year the first car was invented).

- Instantiate an object and try setting/getting values.

Exercise 3: Inheritance

Concepts Covered: Parent & Child Classes, Overriding Methods

- Create a `Vehicle` base class with properties: `$color` and `$speed`.

- Add a method `describe()` that prints a basic description of the vehicle.

- Create a `Bike` class that extends `Vehicle` and adds a property `$type` (e.g., "mountain", "road").

- Override the `describe()` method to include the bike type.

Exercise 4: Polymorphism

Concepts Covered: Method Overriding, Interfaces

- Define an interface `Shape` with a method `getArea()`.

- Create `Rectangle` and `Circle` classes that implement `Shape`.

- Implement the `getArea()` method in both classes (`Rectangle` using width × height, `Circle` using π × r²).

- Instantiate both classes and display their areas.

Exercise 5: Abstract Classes

Concepts Covered: Abstract Classes & Methods

- Create an abstract class `Employee` with properties `$name` and `$salary`.

- Add an abstract method `calculateBonus()`.

- Create two subclasses `Manager` and `Developer`, each implementing `calculateBonus()` with different logic.

- Instantiate both subclasses and calculate bonuses.

Exercise 6: Static Properties & Methods

Concepts Covered: Static Members, Singleton Pattern

- Create a `Logger` class with a static property `$logCount` and a static method `logMessage($message)`.

- Each time `logMessage()` is called, increase `$logCount`.

- Test the logger by logging messages and displaying the count.

- Implement the Singleton Pattern in `Logger` to ensure only one instance exists.

Exercise 7: Magic Methods

Concepts Covered: `__construct()`, `__destruct()`, `__toString()`, `__get()`, `__set()`

- Create a `Person` class with private properties `$name` and `$age`.

- Implement `__construct()` to initialize properties.

- Implement `__get()` and `__set()` to control access.

- Implement `__toString()` to return a string representation of the object.

- Instantiate the class and test these magic methods.

Exercise 8: Dependency Injection

Concepts Covered: Loose Coupling, Constructor Injection

- Create a `Database` class with a `connect()` method.
- Create a `UserRepository` class that requires a `Database` instance in its constructor.
- Implement a method `getAllUsers()` that returns dummy users.
- Instantiate `Database` and inject it into `UserRepository`.

Exercise 9: Traits

Concepts Covered: Code Reusability with Traits

- Create a trait `LoggerTrait` with a method `log($message)`.
- Create two classes, `FileUploader` and `OrderProcessor`, that use `LoggerTrait`.
- Call `log()` in both classes to demonstrate reusability.

Exercise 10: Exception Handling

Concepts Covered: Try-Catch, Custom Exceptions

- Create a custom exception class `InvalidAgeException`.
- Create a function `validateAge($age)` that throws this exception if age is below 18.
- Use `try-catch` to handle the exception.