PHP Object-Oriented Programming Lab Exercise

Week 12

Part 1: Working with Abstract Classes

Problem Description:
Create an abstract class `Animal` with an abstract method `makeSound()`. Implement this abstract class in two subclasses: `Dog` and `Cat`. Each subclass should have its own implementation of `makeSound()`. Create instances of `Dog` and `Cat` and call the `makeSound()` method.

Requirements:

- Abstract class `Animal` with an abstract method `makeSound()`.
- Subclasses `Dog` and `Cat` implementing `makeSound()`.
- Instantiate `Dog` and `Cat` and invoke `makeSound()` to see the output.

Part 2: Utilizing Final Methods

Problem Description:
Create a class `Logger` with a final method `writeLog`. Extend this class with a subclass `FileLogger` and try to override the `writeLog` method. Observe what happens. Then, create an instance of `FileLogger` and call the `writeLog` method.

Requirements:

- Class `Logger` with a final method `writeLog`.
- Subclass `FileLogger` extending `Logger`.
- Attempt to override `writeLog` in `FileLogger`.
- Instantiate `FileLogger` and call `writeLog`.

Part 3: Implementing Interfaces

Problem Description:
Create an interface **`Chargeable`** with a method **`chargeBattery`**. Implement this interface in two classes: **`ElectricCar`** and **`Smartphone`**. Each class should have its own implementation of **`chargeBattery`**. Create instances of **`ElectricCar`** and **`Smartphone`** and call the **`chargeBattery`** method.

Requirements:

- Interface **`Chargeable`** with method **`chargeBattery`**.
- Classes **`ElectricCar`** and **`Smartphone`** implementing **`Chargeable`**.
- Instantiate **`ElectricCar`** and **`Smartphone`** and invoke **`chargeBattery`**.

Part 4: Demonstrating Inheritance

Problem Description:
Create a base class **`Vehicle`** with a method **`startEngine`**. Then, create two subclasses **`Car`** and **`Motorcycle`** that inherit from **`Vehicle`**. Override the **`startEngine`** method in both subclasses. Create instances of **`Car`** and **`Motorcycle`** and call **`startEngine`**.

Requirements:

- Base class **`Vehicle`** with method **`startEngine`**.
- Subclasses **`Car`** and **`Motorcycle`** overriding **`startEngine`**.
- Instantiate **`Car`** and **`Motorcycle`** and call **`startEngine`**.

Part 5: Using `parent` in Constructors

Problem Description:
Modify the **`Vehicle`** class to include a constructor that initializes a **`vehicleType`** property. In **`Car`** and **`Motorcycle`**, use **`parent::__construct()`** to call the parent constructor, then add additional initialization specific to each subclass. Create instances of **`Car`** and **`Motorcycle`** and observe the initialization process.

Requirements:

- **`Vehicle`** class with a constructor initializing **`vehicleType`**.
- Use **`parent::__construct()`** in **`Car`** and **`Motorcycle`** constructors.
- Instantiate **`Car`** and **`Motorcycle`**, observe the initialization.