TP: Python Object-Oriented Programming

Say OL

September 28, 2024

Python Object-Oriented Programming: Problem Set

This problem set is designed to help you practice writing and using Object-Oriented Programming concepts in Python. Follow the instructions for each problem and document your progress.

Problem 1: Class Definition

a. Define a class called `Car` with attributes: `make`, `model`, and `year`.

b. Include a method `get_description` that returns the car's details in the format: "Year Make Model".

Problem 2: Creating Objects

a. Create an object of the `Car` class called `my_car` with the following values:

- Make: "Toyota"
- Model: "Corolla"
- Year: 2020

b. Print the description of `my_car` using the `get_description` method.

Problem 3: Encapsulation

a. Modify the `Car` class to include private attributes for `make`, `model`, and `year`.

b. Create getter and setter methods for each attribute to allow controlled access to these private variables.

Problem 4: Inheritance

a. Define a new class called `ElectricCar` that inherits from the `Car` class.

b. Add a new attribute `battery_size` and a method `get_battery_size` that returns the battery size.

Problem 5: Method Overriding

a. Override the `get_description` method in the `ElectricCar` class to include the battery size in the description.

Problem 6: Polymorphism

a. Create a function `print_car_description` that takes a `Car` object as an argument and prints its description.

b. Call this function with both a `Car` object and an `ElectricCar` object.

Problem 7: Abstract Classes

a. Create an abstract class `Animal` with an abstract method `sound`.

b. Define two derived classes, `Dog` and `Cat`, that implement the `sound` method.

Problem 8: Composition

a. Create a class `Owner` that has attributes `name` and `pet` (an instance of `Dog` or `Cat`).

b. Include a method `get_pet_sound` that returns the sound of the pet.

Problem 9: Operator Overloading

a. Create a class `Point` representing a point in 2D space.

b. Overload the + operator to allow adding two `Point` objects together.

Problem 10: Design Patterns - Singleton

a. Implement a Singleton pattern for a `Logger` class that logs messages to the console, ensuring only one instance exists.

Submission

Submit your Python files containing the class implementations and any necessary test scripts or documentation of your progress.