SABUDH

Topic: Object-Oriented Programming using Python

Objective: The objective of this assignment is to assess the student's understanding of object-oriented programming concepts and their ability to apply these concepts using Python.

Instructions:
1. The assignment will consist of <n> coding problems related to object-oriented Programming using Python.
2. The student is required to write a complete Python code, along with comments explaining the approach taken to solve the problem.<n>
3. The code should be well-structured and easy to read, with appropriate variable names and indentation.
4. The student is allowed to use any built-in Python libraries but is not allowed to use any third-party libraries.
5. The code should be submitted in a single Python file (.py) along with a brief explanation of the code and its output.

Task 1: Creating a Class (Abstraction)

Create an abstract Python class called Person that has the following attributes:

- Name
- Age
- Gender
- Address

- The class should implement cooperative inheritance
  Hint: cooperative inheritance: Use super() to ensure that methods from all parent classes are called properly in accordance with Python's Method Resolution Order (MRO).
- Define the magic str method that returns the basic info about the person
- Define a method greet that accepts an instance of the Person class and greets the person, e.g., the output should look like Hello John! My name is Jane.
- Define an abstract method that must be implemented by the child classes

- Define a static method is_adult that accepts an argument age and returns True or False if the person is above 18 years old.

Task 2: Single Inheritance, Encapsulation

Create a Python class called Employee that inherits from the Person class created in Problem

1. This class must also implement cooperative inheritance
   Hint: cooperative inheritance: Use super() to ensure that methods from all parent classes are called properly in accordance with Python's Method Resolution Order (MRO).

- The Employee class should have the following attributes:
- Create a class attribute counter that will increase by one when a new instance of
- The employee is initialised and decreased by one when an instance is deleted, a private attribute employee_id that holds the value according to the counter, e.g., EMP01, EMP02. The private attribute must have only the getter method, not the setter; employee_id should not be able to be changed once it is created. A protected salary attribute

2. The class should have the following methods:

- A constructor that initialises the attributes.
- A method called counter wrapped in the property decorator that returns the class variable counter
- Getter and setter methods for salary, and also methods that increase and decrease the salary.
- An introduction method that overrides the abstract method defined in the Person class.

Task 3: Multiple Inheritance, Polymorphism

Create a Python class called Teacher that inherits from the Employee, Person classes created
in Problems 1 and 2. This class must also implement cooperative inheritance
Hint: cooperative inheritance: Use super() to ensure that methods from all parent classes are called properly in accordance with Python's Method Resolution Order (MRO).

The Teacher class should have the following attributes:

- Create a class attribute counter that will increase by one when a new instance of the employee is initialised and decrease by one when an instance is deleted
- A private attribute teacher_id that holds the value according to the counter, e.g., TEC01, TEC02. The private attribute must have only the getter method, not the setter; teacher_id should not be able to be changed once it is created
- A subject's attribute is a list of subjects.

The class should have the following methods:

- A constructor that initialises the attributes.
- A method called counter wrapped in the property decorator that returns the class variable counter methods that append or remove a particular subject from the subjects list.
- An introduction method that overrides the abstract method defined in the Person class that should return the teacher_id and the list of subjects.
- Since we have an attribute named teacher_id, we won't need employee_id; override the employee_id that now returns an AttributeError if someone tries to access the attribute employee_id. E.g.Teacher object has no attribute employee ID.

Note: the class name Teacher must not be hardcoded. It should be dependent on the class name so that if a new child class inherits from the Teacher class, it should not again say Teacher.

Grading: Your program will be graded based on the following criteria:

1. Correct implementation of the OOP concepts
2. Proper implementation of methods and attributes.
3. Proper output from the methods.
4. Proper indentation and naming conventions
5. All parent class attributes must be correctly inherited by the children
6. The classes must follow cooperative inheritance
7. Proper public, private and protected attributes implementation
8. Proper implementation of abstraction and static methods
9. Proper implementation of magic methods
10. The method resolution order should be correct: Teacher > Employee > Person > ABC > Object

Good Luck!