PYTHON

Assignment

1. Create a class `Person` with a private attribute `_age`. Implement a method `get_age()` to access the value of `_age` and `set_age()` to modify the value while keeping the attribute private.
2. Create a class `Employee` with a private attribute `__salary`. Verify how Python handles name mangling by accessing the private attribute using `_Employee__salary`.
3. Implement a class `Product` with a private attribute `_price`. Create getter and setter methods using Python's property decorator to access and modify the price with validation (e.g., price can't be negative).
4. Define a class `Circle` with a private attribute `_radius` and create a read-only property `radius` that allows getting but not setting the value directly.
5. Create a class `BankAccount` with both a protected attribute `_balance` and a private attribute `__pin`. Implement methods to access and modify both attributes, explaining the difference between protected and private access.
6. Write a class `Student` with a private attribute `_grade`. Use the setter method to ensure that the grade is between 0 and 100.
7. Create a class `Temperature` with a private attribute `_celsius`. Implement getter and setter methods that convert between Celsius and Fahrenheit (e.g., set in Fahrenheit, store in Celsius).
8. Implement a `BankAccount` class with private attributes `_account_number` and `_balance`. Create methods to deposit and withdraw money, ensuring that the balance cannot become negative.
9. Write a class `Rectangle` with private attributes `_length` and `_width`. Use getter and setter methods for each, but only allow the setter for `width` to modify the attribute if the value is greater than zero.
10. Design a class `Vehicle` with private attributes `_speed` and `_fuel_level`. In the constructor, ensure that both attributes are set with valid values using encapsulation principles. Include getter and setter methods to access and modify the attributes.