Assignment 5

5
Aim: Create a Shape superclass with a method to calculate the area, and Circle and Rectangle subclasses that override the area calculation method to demonstrate polymorphism. Create a Book class with private attributes for title, author, and ISBN, and public getter and setter methods to access and modify these attributes, demonstrating encapsulation.
Code:

```
import math

# Shape superclass to demonstrate polymorphism
class Shape:
    def __init__(self, name="Shape"):
        self.name = name

    def calculate_area(self):
        """Base method to calculate area of a shape"""
        return 0

    def __str__(self):
        return f"{self.name} with area: {self.calculate_area()}"

# Circle subclass
class Circle(Shape):
    def __init__(self, radius):
        super().__init__("Circle")
        self.radius = radius

    def calculate_area(self):
        """Override area calculation for circle"""
        return math.pi * self.radius ** 2

# Rectangle subclass
class Rectangle(Shape):
    def __init__(self, length, width):
        super().__init__("Rectangle")
        self.length = length
        self.width = width

    def calculate_area(self):
        """Override area calculation for rectangle"""
        return self.length * self.width

# Book class to demonstrate encapsulation
```

```
class Book:
    def __init__(self, title, author, isbn):
        # Private attributes using double underscore
        self.__title = title
        self.__author = author
        self.__isbn = isbn

    # Getter methods
    def get_title(self):
        return self.__title

    def get_author(self):
        return self.__author

    def get_isbn(self):
        return self.__isbn

    # Setter methods
    def set_title(self, title):
        self.__title = title

    def set_author(self, author):
        self.__author = author

    def set_isbn(self, isbn):
        if len(isbn) == 13 or len(isbn) == 10:  # Basic validation for
ISBN
            self.__isbn = isbn
        else:
            print("Invalid ISBN format")

    def __str__(self):
        return f"Book: {self.__title} by {self.__author}, ISBN:
{self.__isbn}"


# Test code to demonstrate the classes
if __name__ == "__main__":
    # Polymorphism demonstration
    print("POLYMORPHISM DEMONSTRATION")
    shapes = [Circle(5), Rectangle(4, 6), Shape()]

    for shape in shapes:
        print(shape)
```

```python
# Encapsulation demonstration
print("\nENCAPSULATION DEMONSTRATION")
book = Book("Python Programming", "John Smith", "9781234567890")
print(book)

# Accessing private attributes through getter methods
print(f"Title: {book.get_title()}")
print(f"Author: {book.get_author()}")
print(f"ISBN: {book.get_isbn()}")

# Modifying private attributes through setter methods
book.set_title("Advanced Python Programming")
book.set_author("Jane Doe")
book.set_isbn("1234567891012")
print("\nAfter modification:")
print(book)
```

Output Screenshot:

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS    GITLENS    COMMENTS

python3 -u "/Users/debdootmanna/VSCode/Python/Assignment 5.py"
❯ python3 -u "/Users/debdootmanna/VSCode/Python/Assignment 5.py"
POLYMORPHISM DEMONSTRATION
Circle with area: 78.53981633974483
Rectangle with area: 24
Shape with area: 0

ENCAPSULATION DEMONSTRATION
Book: Python Programming by John Smith, ISBN: 9781234567890
Title: Python Programming
Author: John Smith
ISBN: 9781234567890

After modification:
Book: Advanced Python Programming by Jane Doe, ISBN: 1234567891012

~/VSCode/Python  on  main ?4
❯
```

Conclusion/Summary:

This assignment demonstrates two fundamental principles of object-oriented programming:

Polymorphism: The Shape hierarchy shows how different subclasses (Circle and Rectangle) can implement the same method (calculate_area()) in different ways while maintaining a consistent interface. This allows for flexible code that can work with various shapes without needing to know their specific implementations.

Encapsulation: The Book class illustrates how to hide implementation details by making attributes private and providing controlled access through getter and setter methods. This protects the data integrity (as shown by the ISBN validation) and creates a clean, stable interface for other code to interact with.

Student Signature & Date	**Marks:**	**Evaluator Signature & Date**