

Laboratory Activity No. 7

Polymorphism

Course Code: CPE103

Program: BSCPE

Course Title: Object-Oriented Programming

Date Performed: 02 / 28 / 2025

Section: 1-A

Date Submitted: 02 / 28 / 2025

Name: Asugas, Kenneth R.

Instructor: Engr. Maria Rizette Sayo

1. Objective(s):

This activity aims to familiarize students with the concepts of Polymorphism in Object-Oriented Programming

2. Intended Learning Outcomes (ILOs):

The students should be able to:

2.1 Identify the use of Polymorphism in Object-Oriented Programming

2.2 Implement an Object-Oriented Program that applies Polymorphism

3. Discussion:

Polymorphism is a core principle of Object-Oriented that is also called "method overriding". Simply stated the principles says that a method can be redefined to have a different behavior in different derived classees.

For an example, consider a base file reader/writer class then three derived classes Text file reader/writer, CSV file reader/ writer, and JSON file reader/writer. The base file reader/writer class has the methods: read(filepath="") , write(filepath=""). The three derived classes (classes that would inherit from the base class) should have behave differently when their read, write methods are invoked.

Operator Overloading:

Operator overloading is an important concept in object oriented programming. It is a type of polymorphism in which a user defined meaning can be given to an operator in addition to the predefined meaning for the operator.

Operator overloading allow us to redefine the way operator works for user-defined types such as objects. It cannot be used for built-in types such as int, float, char etc., For example, '+' operator can be overloaded to perform addition of two objects of distance class.

Python provides some special function or magic function that is automatically invoked when it is associated with that particular operator. For example, when we use + operator on objects, the magic method __add__() is automatically invoked in which the meaning/operation for + operator is defined for user defined objects.

4. Materials and Equipment:

Windows Operating System
Google Colab

5. Procedure:

Creating the Classes

1. Create a folder named oopfa1<lastname>_lab8
2. Open your IDE in that folder.
3. Create the base polymorphism_a.ipynb file and Class using the code below:

Coding:

distance is a class. Distance is measured in terms of feet and inches

```
class distance:
```

```
    def __init__(self, f,i):
```

```
        self.feet=f
```

```
        self.inches=i
```

overloading of binary operator > to compare two distances

```
    def __gt__(self,d):
```

```
        if(self.feet>d.feet):
```

```
            return(True)
```

```
        elif((self.feet==d.feet) and (self.inches>d.inches)):
```

```
            return(True)
```

```
        else:
```

```
            return(False)
```

overloading of binary operator + to add two distances

```
    def __add__(self, d):
```

```
        i=self.inches + d.inches
```

```
        f=self.feet + d.feet
```

```
        if(i>=12):
```

```
            i=i-12
```

```
            f=f+1
```

```
        return distance(f,i)
```

displaying the distance

```
    def show(self):
```

```
        print("Feet= ", self.feet, "Inches= ",self.inches)
```

```
a,b= (input("Enter feet and inches of distance1: ")).split()
```

```
a,b =[int(a),int(b)]
```

```
c,d= (input("Enter feet and inches of distance2: ")).split()
```

```
c,d =[int(c),int(d)]
```

```
d1 = distance(a,b)
```

```
d2 = distance(c,d)
```

```
if(d1>d2):
```

```
    print("Distance1 is greater than Distance2")
```

```
else:
```

```
    print("Distance2 is greater or equal to Distance1")
```

```
d3=d1+d2
```

```
print("Sum of the two Distance is:")
```

```
d3.show()
```

4. Screenshot of the program output:

```
Feet= 5 Inches= 10
Enter feet and inches of distance1: 10 20
Enter feet and inches of distance2: 30 40
Distance2 is greater or equal to Distance1
Sum of the two Distance is:
Feet= 41 Inches= 48
Enter feet and inches of distance1: 123 52
Enter feet and inches of distance2: 69 147
Distance1 is greater than Distance2
```

Testing and Observing Polymorphism

1. Create a code that displays the program below:

```
class RegularPolygon:
    def __init__(self, side):
        self._side = side
class Square (RegularPolygon):
    def area (self):
        return self._side * self._side
class EquilateralTriangle (RegularPolygon):
    def area (self):
        return self._side * self._side * 0.433

obj1 = Square(4)
obj2 = EquilateralTriangle(3)

print (obj1.area())
print (obj2.area())
```

2. Save the program as polymorphism_b.ipynb and paste the screenshot below:

```
class RegularPolygon:
    def __init__(self, side):
        self._side = side

class Square(RegularPolygon):
    def area(self):
        return self._side * self._side

class EquilateralTriangle(RegularPolygon):
    def area(self):
        return self._side * self._side * 0.433

obj1 = Square(4)
obj2 = EquilateralTriangle(3)

print(obj1.area())
print(obj2.area())
```

16
3.897

3. Run the program and observe the output.
4. Observation:

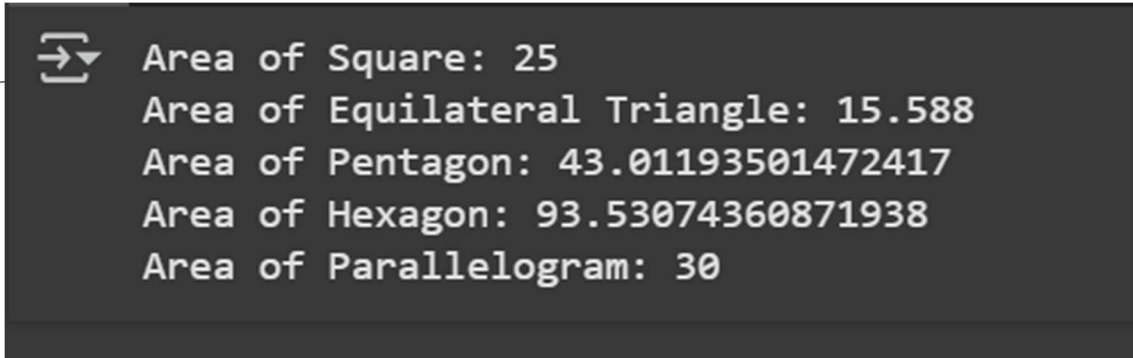
When I tried running the code I found that both the Square and Equilateral Triangle classes were inheriting from the Regular Polygon base class. Even though they have the same method name area(), each class calculates the area in a different way based on the shape of the object.

The Square class multiplies the side by itself; the Equilateral Triangle class multiplies the side by itself plus some constant (0.433). So Polymorphism: calculating area in a different way depending on what class the object belongs to.

For the program please refer to this link: https://github.com/Kenneth-Asugas/CPE-103-OOP-1-A/blob/main/Lab_7.ipynb

6. Supplementary Activity:

In the above program of a Regular polygon, add three more shapes and solve for their area using each proper formula. Take a screenshot of each output and describe each by typing your proper labeling.



```
→ Area of Square: 25
Area of Equilateral Triangle: 15.588
Area of Pentagon: 43.01193501472417
Area of Hexagon: 93.53074360871938
Area of Parallelogram: 30
```

For the program please refer to this link: https://github.com/Kenneth-Asugas/CPE-103-OOP-1-A/blob/main/Lab_7.ipynb

1. RegularPolygon is the basic class that initializes the side length of a regular polygon. It is the backbone class for all other polygon classes.
2. Square is a derived class that represents a square (with all sides symmetrical); area is defined by the expression (side * side) multiplied by side.
3. EquilateralTriangle is a derived class of triangle that represents an equilateral triangle in which the whole triangle is an equal length rectangle and the area is approximated with the formula (side squared multiplied by 0. 433) derived from the square root of (3*4).
4. Pentagon is a derived class that represents a regular pentagon (five sided polygon), and the area is calculated via a mathematical formula (involving the square root of five and related terms) for such a pentagon.
5. Hexagon is a derived class that represents a regular hexagon, a six-sided polygon. The area is calculated using the formula three times the square root of three divided by two, multiplied by side squared.
6. Parallelogram is a derived class that represents a parallelogram (a quadrilateral whose two opposite sides are parallel)and has other attributes for base and height; the area is calculated by the following formula: base * height.

Every one of these polygon classes is instantiated as objects and the area of each shape is printed using the area method.

Questions

1. Why is Polymorphism important?

Polymorphism means that one interface can contain a whole bunch of different types of data (flexibility), allowing code to be written easier, cleaner and more scalable.

2. Explain the advantages and disadvantages of using applying Polymorphism in an Object-Oriented Program.

Advantages: Polymorphism makes code more flexible, reuseable and easy to maintain.

Disadvantages: Code complexity may increase and it may result in performance overhead because of dynamic binding of methods.

3. What maybe the advantage and disadvantage of the program we wrote to read and write csv and json files?

Advantage: The program can work with different file formats with the same code, which is a huge advantage and reduces redundancy.

Disadvantage: Intangible complexity of code may make it harder to understand and maintain, Polymorphic behavior may not be as fast as discrete behaviors.

4. What maybe considered if Polymorphism is to be implemented in an Object-Oriented Program?

Generally good design patterns should be used to keep the code organized and maintainable. Performance impact and significant testing is needed to make sure the behavior is correct as well as good documentation of code for other developers.

5. How do you think Polymorphism is used in an actual programs that we use today?

Polymorphism is used in UIs to manipulate different types of user inputs, and in data processing libraries such as Pandas to manipulate different data formats. It 's also used in game development to manipulate different entities such as players, enemies, and objects.

7. Conclusion:

Polymorphism is a principle of object oriented programming that has been proved to greatly enhance the flexibility, reuse, and maintainability of code. It may lead to complexity and/or performance disadvantages with code, but the many benefits of polymorphism in an attempt at promoting a scalable and modular design often outweigh any disadvantages. The knowledge and implementation of polymorphism is necessary to make a software system strong and efficient.

8. Assessment Rubric: