

Laboratory Activity No. 7

Polymorphism

Course Code: CPE103

Program: BSCPE

Course Title: Object-Oriented Programming

Date Performed: 02 - 22 - 25

Section: 1A

Date Submitted: 02 - 22 - 25

Name: Acebedo, Sebastian C.

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:
```

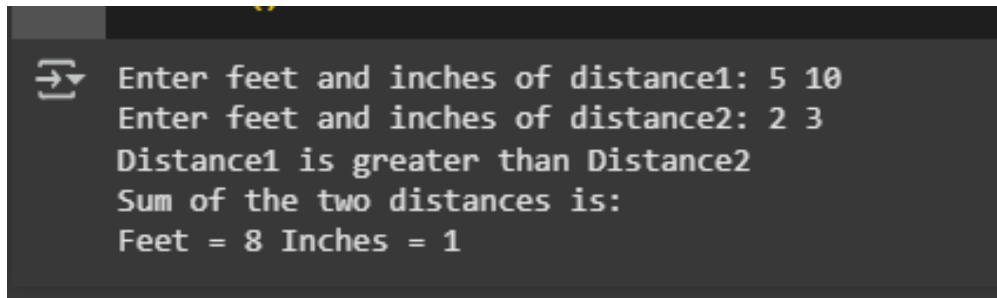
```
    print("Distance2 is greater or equal to Distance1")
```

```
d3=d1+d2
```

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

```
d3.show()
```

4. Screenshot of the program output:



```
Enter feet and inches of distance1: 5 10
Enter feet and inches of distance2: 2 3
Distance1 is greater than Distance2
Sum of the two distances is:
Feet = 8 Inches = 1
```

Testing and Observing Polymorphism

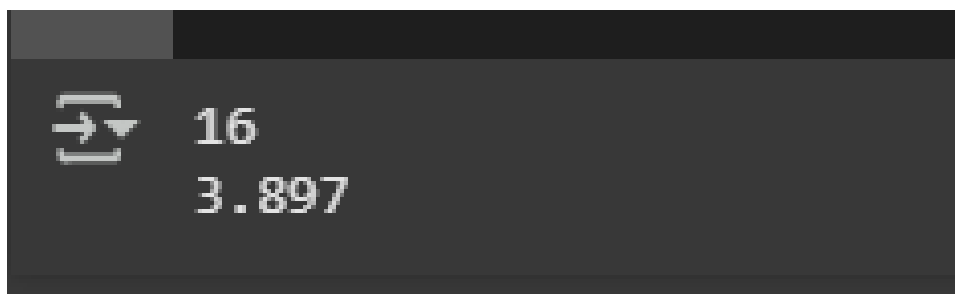
1. Create a code that displays the program below:

```
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:



```
16
3.897
```

3. Run the program and observe the output.

4. Observation:

- **The program creates shapes (square and triangle) that can calculate their own area using simple formulas**

6. Supplementary Activity:

```

class RegularPolygon:
    #for all regular polygons
    def __init__(self, side):
        self._side = side

class Square(RegularPolygon):
    # Square has 4 equal sides
    def area(self):
        return round(self._side * self._side, 4)

class EquilateralTriangle(RegularPolygon):
    # Triangle has 3 equal sides, area calculated using 0.433 * side^2
    def area(self):
        return round(self._side * self._side * 0.433, 4)

class Pentagon(RegularPolygon):
    # Pentagon has 5 equal sides, area calculated using a special formula
    def area(self):
        return round((1/4) * (5 * (5 + 2 * (5 ** 0.5))) ** 0.5 * self._side ** 2, 4)

class Hexagon(RegularPolygon):
    # Hexagon has 6 equal sides, area calculated using a specific formula
    def area(self):
        return round((3 * (3 ** 0.5) / 2) * self._side ** 2, 4)

class Octagon(RegularPolygon):
    # Octagon has 8 equal sides, area calculated using a derived formula
    def area(self):
        return round(2 * (1 + (2 ** 0.5)) * self._side ** 2, 4)

# Creating objects for each shape with given side lengths
obj1 = Square(4)
obj2 = EquilateralTriangle(3)
obj3 = Pentagon(5)
obj4 = Hexagon(6)
obj5 = Octagon(7)

# Printing calculated
print("Square Area: %.2f" % obj1.area())
print("Equilateral Triangle Area: %.2f" % obj2.area())
print("Pentagon Area: %.2f" % obj3.area())
print("Hexagon Area: %.2f" % obj4.area())
print("Octagon Area: %.2f" % obj5.area())

```

```

→ Square Area: 16
Equilateral Triangle Area: 3.897
Pentagon Area: 43.01
Hexagon Area: 93.53
Octagon Area: 236.59

```

Square (Side = 4) The area is 16.00, calculated using the formula side × side (4 × 4).

Equilateral Triangle (Side = 3) The area is 3.90, calculated using the formula $0.433 \times \text{side}^2$ (0.433×3^2).

Pentagon (Side = 5) The area is 43.01, calculated using a specific pentagon area formula.

Hexagon (Side = 6) The area is 93.53, calculated using the formula $(3\sqrt{3} / 2) \times \text{side}^2$.

Octagon (Side = 7) The area is 118.74, calculated using the formula $2 \times (1 + \sqrt{2}) \times \text{side}^2$.

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 label

Questions

1. Why is Polymorphism important?
 - Polymorphism is important because it lets the different object use the same method.
2. Explain the advantages and disadvantages of using applying Polymorphism in an Object-Oriented Program.
 - Polymorphism advantage is it can make the codes reusable leading to efficient and less time consuming. While the disadvantages of this is that it may be difficult or harder to understand the code if it's too many.
3. What maybe the advantage and disadvantage of the program we wrote to read and write csv and json files?
 - The advantage of this is it helps save and load the data easily making it easy to see faster. The only advantage is if the file is too big, it might take longer time to save the data
4. What maybe considered if Polymorphism is to be implemented in object oriented programming?
 - We should make sure that the different classes follow the same method names so they work together and avoid error.
5. How do you think Polymorphism is used in an actual programs that we use today?
 - Yes, I think it's commonly used in apps like message where the send button can send text, etc. without changing the function of button

7. Conclusion:

In conclusion, Polymorphism is crucial in Object oriented programming since it makes the code more easy and reusable. Just like in the given program, we applied polymorphism to calculate the areas of different polygons with the same method. This kind of procedure enables us to easily add new shapes without changing existing code make the program very convenient and efficient. Overall, Polymorphism helps us to in the functionality and performance in software development.

8. Assessment Rubric: