Laboratory Activity No. 7	
Polymorphism	
Course Code: CPE103	**Program:** BSCPE
Course Title: Object-Oriented Programming	**Date Performed:** 02/22/25
Section: 1-A	**Date Submitted:** 02/22/25
Name: Villanueva, Bryan O.	**Instructor:** Engr. Maria Rizette H. 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 distance 1: 5 8
Enter feet and inches of distance 2: 4 7
Distance 1 is greater than distance 2
The sum of the distances is:
Feet =  10 Inches =  3
```

Testing and Observing Polymorphism

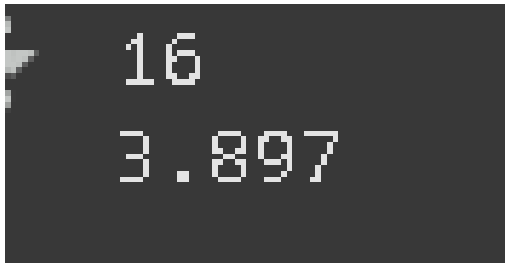
1. Create a code that displays the program below:

```
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:



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

PLEASE REFER TO THIS LINK FOR MY ANSWER:
CODES
https://colab.research.google.com/drive/1OEUtvJhAQoFW8Jgc_Ls2rJ0PGbEY-6d3?authuser=2#scrollTo=p-lzqpKXTv-Q

OBSERVATION
https://colab.research.google.com/drive/1OEUtvJhAQoFW8Jgc_Ls2rJ0PGbEY-6d3?authuser=2#scrollTo=83uEpW-O8_mv&line=1&uniqifier=1

BELOW

FOR MY SUPPLEMENTARY ACTIVITY PLEASE REFER TO THIS LINK
https://colab.research.google.com/drive/1OEUtvJhAQoFW8Jgc_Ls2rJ0PGbEY-6d3?authuser=2#scrollTo=YLNah94LaPJA

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

Supplementary Activity

```python
import math

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

# Pentagon class (5 sides)
class Pentagon(RegularPolygon):
    def area(self):
        return (5 / 4) * (self._side ** 2) * (1 / math.tan(math.pi / 5))

# Hexagon class (6 sides)
class Hexagon(RegularPolygon):
    def area(self):
        return (3 * math.sqrt(3) / 2) * (self._side ** 2)

# Octagon class (8 sides)
class Octagon(RegularPolygon):
    def area(self):
        return 2 * (1 + math.sqrt(2)) * (self._side ** 2)


obj1 = Pentagon(5)
obj2 = Hexagon(6)
obj3 = Octagon(8)


print(f"Area of Pentagon: {obj1.area():.2f}")
print(f"Area of Hexagon: {obj2.area():.2f}")
print(f"Area of Octagon: {obj3.area():.2f}")
```

```
Area of Pentagon: 43.01
Area of Hexagon: 93.53
Area of Octagon: 309.02
```

.

SUPPLEMENTARY:

Pentagon consists of 5 sides, and I used the area of pentagon formula to calculate the area of this regular polygon. After calculating, I got 43.01 for the area.

Hexagon has 6 sides, and I also used the formula for this regular polygon to get the area. After calculating, I got 93.53 for the area.

Octagon has 8 sides, and to get the exact area of this regular polygon, I needed to use the area formula for this shape. After calculating, I got 309.02 for the area.

Questions

1. Why is Polymorphism important?

- Code may treat distinct classes consistently because to polymorphism, which gives objects the ability to take on multiple forms.

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

- Code reusability is increased by polymorphism, but readability is decreased and implementation might be challenging.

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

- Managing several file types using a single interface is advantageous; addressing file-specific subtleties becomes more difficult.

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

- When implementing polymorphism, think about the shared interface objects and the effects of runtime method resolution on performance.

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

- When a single function can handle many object types, polymorphism is utilized in real programs to simplify code and facilitate program extension.

7. Conclusion:

To sum up, polymorphism improves code flexibility but necessitates cautious application and evaluation of any potential disadvantages.

8. Assessment Rubric: