| Laboratory Activity No. 7 | |
|---|---|
| **Polymorphism** | |
| **Course Code:** CPE103 | **Program:** BSCPE |
| **Course Title:** Object-Oriented Programming | **Date Performed:** February 22, 2025 |
| **Section:** 1-A | **Date Submitted:** March 4, 2025 |
| **Name:** Directo, Hannah Thea B. | **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 classes.

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

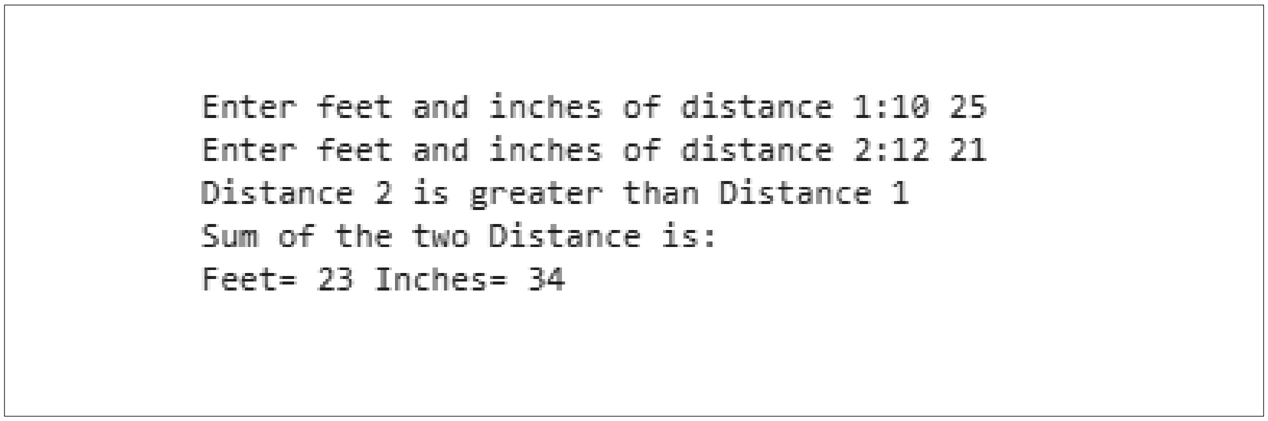
## 5. Procedure:

**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:



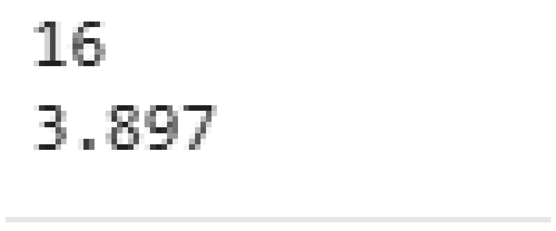
```
Enter feet and inches of distance 1:10 25
Enter feet and inches of distance 2:12 21
Distance 2 is greater than Distance 1
Sum of the two Distance is:
Feet= 23 Inches= 34
```

**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
```

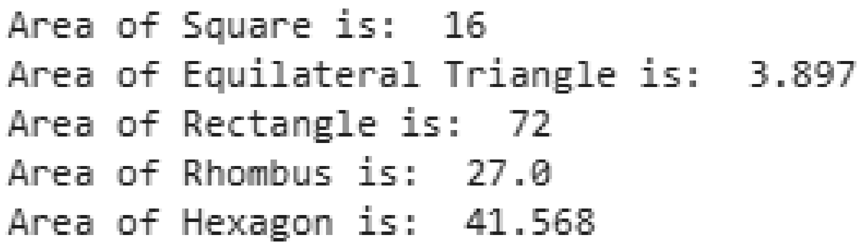
3. Run the program and observe the output.

4, Observation

The code shows a concept of polymorphism by defining a class called RegularPolygon, along with two subclasses, Square and EquilateralTriangle. Each subclasses have their own calculation for area using the area() Method. Which means that the program calculate the area of triangle and equilateral triangle differently because they have different formula. In this concept, different types of objects can use the same method in their own way.

**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 is:  16
Area of Equilateral Triangle is:  3.897
Area of Rectangle is:  72
Area of Rhombus is:  27.0
Area of Hexagon is:  41.568
```

Polymorphism.ipynb - Colab

1. **Regular Polygon:** Serves as the basic template for any shapes that has sides. It stores the length of one side referred as _side. It is like the parent of all the shapes.
2. **Square:** This class Inherits from RegularPolygon and represents a square. It calculates the area using the formula : side^2
3. **Equilateral Triangle:** It inherits also from RegularPolygon, it represents equilateral triangle (all sides are equal). To calculate the area we used the approximation: side*side*0.433
4. **Rectangle:** This class represents a rectangle. It needs two sides–side 1 side 2 (length and width). By multiplying the width and length, the area will be calculated. This class is considered as part of the RegularPolygon even though the sides are not equal to each other.
5. **Rhombus**: This class represents a rhombus, which is a special type of quadrilateral where all four sides are of equal length. However, instead of using the sides to define it, we use two diagonals (the lines that connect opposite corners). This class is still part of the RegularPolygon group. To calculate the area we multiply the lengths of both diagonals and then divide the result by 2.
6. **Hexagon:** This class represents a hexagon, which is a shape with six sides. If all six sides are equal, it is called a **regular hexagon**. This class is part of the **RegularPolygon** group. To calculate the area of hexagon we take the square of the side length, multiply it by **3√3** or 1.732 and then divide by 2.

**Questions**

1. Why is Polymorphism important?

Polymorphism is important because it allows different types of objects to be treated as if they are the same. This makes programming easier and more flexible.

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

The advantage of Polymorphism, it makes the code easier to manage and reuse. Instead of writing the same code multiple times for different objects we can write it once and use it for another type. While the disadvantage of it is it makes the code more complex and harder to debug, especially if many objects behave differently but share the same method same names.

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

The program can handle both using the same approach, making it easier to work with different types. This saves time and effort. The disadvantage is that dealing with different structures can be complicated, and there may be extra steps needed to correctly format the data when converting between CSV and JSON.

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

Before using polymorphism, it's important to make sure that different objects share a common relationship, Also, we need to ensure that when a method is overridden, it still works, If not used properly, it might lead to errors or unexpected results.

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

Polymorphism is used in graphical interfaces (GUIs), buttons, text fields, and drop-down menus all behave differently but are treated As common UI elements. In web applications, different types of HTTP requests (such as GET, POST, and DELETE) use the same Interface but perform different tasks.

## 7.Conclusion

Polymorphism is tool in object-oriented programming that makes the program more flexible and reusable that allows different classes to use the same method in their own way. This reduces code repetition and makes the program easier to update and expand. The example code shows how polymorphism enables different shapes to calculate their areas through a shared interface. Polymorphism makes programs more adaptable and scalable, which is essential for real-world applications.

## 8. Assessment

**Questions**

6. Why is Polymorphism important?

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

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

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

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

## 7. Conclusion:

## 8. Assessment Rubric: