Laboratory Activity No. 7	
Polymorphism	
Course Code: CPE103	**Program:** BSCPE
Course Title: Object-Oriented Programming	**Date Performed: 2025-2-22**
Section: 1A	**Date Submitted:**
Name: Ruperto, April Anne A	**Instructor: 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 distance1: 5 6
Enter feet and inches of distance2: 1 2
Distance1 is greater than Distance2
Sum of the two Distance is:
Feet= 6 Inches= 8
```

Testing and Observing Polymorphism

1. Create a code that displays the program below:

```python
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 specifies a parent class RegularPolygon with a side field and two child classes, Square and EquilateralTriangle, that inherit from RegularPolygon and specify their own area methods to calculate their own respective areas. It then constructs instances of Square and EquilateralTriangle, obj1 and obj2, with certain side lengths and invokes their area methods to print the computed areas. This illustrates inheritance in object-oriented programming, where code reuse and specialized behavior in child classes are enabled.

6. Supplementary Activity:

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

```python
import math

class RegularPolygon:
  def __init__ (self, vertices):
    self._vertices = vertices

#Defining the Rectangle Class
class Rectangle (RegularPolygon):
  def __init__(self, length, width):
    self.length = length
    self.width = width

  def area (self):
    return self.length * self.width

#Defining the Circle Class
class Circle (RegularPolygon):
  def __init__(self, radius):
    self.radius = radius

  def area (self):
    return math.pi * self.radius * self.radius

#Defining the Pentagon Class
class Pentagon (RegularPolygon):
  def area (self):
    return 1/2 * math.sqrt(5 * (5 + 2 * math.sqrt(5))) * self._vertices * self._vertices

#Creating Objects
obj1 = Rectangle(4, 5)
obj2 = Circle(3)
obj3 = Pentagon(5)

#Output
print (obj1.area())
print (obj2.area())
print (obj3.area())
```

```
20
28.274333882308138
86.02387002944835
```

Please refer to this link: [CPE-103-OOP-1A/OOP1Ruperto_lab8.ipynb at main · Ruperto-April-Anne/CPE-103-OOP-1A]

Questions

1. Why is Polymorphism important?

- Polymorphism is something that allows objects that are using different types to be treated as instances of the common superclass. This makes the code reusable, which is the biggest advantage. Also, it provides qualities like flexibility, and scalability by the possibility to bind methods and classes to the objects of the different types through a single interface, improving maintainability and reducing complexity.

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

- Polymorphism improves code efficiency, flexibility, and sustainability which allows the same method to be used for different object types. However, it may also cause some issues with the complexity and debugging that come with dynamic method resolution especially with deep inheritance structures.

3. What may be the advantages and disadvantages of the program we wrote to read and write csv and json files?

- One of the pros of the applications in how to deal with instigating and integrating the CSV and the JSON files is the human range and visibility that it is easy to read and to handle with. At the same time, there may be some problems with large data that can be solved with a better understanding of strong typing, or it involves errors that may occur with an odd number or the mixture or varying data structure.

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

- When it comes to polymorphism, the preference for the use of interfaces or abstract classes, the alignment of the method signatures and behaviors and the estimation of performance which it may influence, are some of the things to put emphasis on. Testing is critical to avoid unexpected behavior, especially in dynamic dispatch scenarios.

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

- Polymorphism is widely used in GUIs, game development, web frameworks, and data processing libraries, allowing for flexible and dynamic handling of different object types or actions through common interfaces, simplifying code management and enhancing system extensibility.

7. Conclusion:

Polymorphism enhances code reusability, flexibility, and scalability by allowing different object types to be managed through a common interface. While it simplifies code and maintenance, it can add complexity, especially with deep inheritance and dynamic method resolution. Effective use requires attention to method consistency, performance, and testing. In real-world applications like GUIs and web frameworks, polymorphism enables dynamic handling of objects, improving system extensibility and management.

8. Assessment Rubric: