

Laboratory Activity No. 7

Polymorphism

Course Code: CPE103

Program: BSCPE

Course Title: Object-Oriented Programming

Date Performed: February 23, 2025

Section: 1-A

Date Submitted: February 23, 2025

Name: Polestico, Paul Justine D.

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: 15 180
Enter feet and inches of distance2: 20 240
Distance2 is greater or equal to Distance1
Sum of the two Distance is:
Feet= 36 Inches= 408
```

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:



```
16
3.897
```

This program calculates the area of the Regular Polygons. In the code above, polygons differ in terms of shapes,number of sides,and their computations, but share the same characteristic, which is the sides.

```
56.02791308792039
```

```
#Calculates the area of a Regular Pentagon
```

```
178.066
```

```
#Calculates the area of a Regular Heptagon
```

308.99199999999996

```
#Calculates the area of a Regular Octagon
```

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.

Questions

1. Why is Polymorphism important?
If objects have similar characteristics, like the RegularPolygon superclass, this can be distributed while also differentiating them in terms of number of sides and computation methods.
2. Explain the advantages and disadvantages of using applying Polymorphism in an Object-Oriented Program.
Polymorphism helps in making the code shorter, provided they have a similar characteristic relative to each other. If not, it could prove more tedious to fix the code if errors are present.
3. What maybe the advantage and disadvantage of the program we wrote to read and write csv and json files?
From my observation, the properties read and write can be distributed to these two, but differ in behavior. However, these similarities may cause confusion, especially in debugging.
4. What maybe considered if Polymorphism is to be implemented in an Object-Oriented Program?
If implementing Polymorphism in your program, objects with one or more similar characteristics to each other must be considered.
5. How do you think Polymorphism is used in an actual programs that we use today?
Polymorphism is used in programs to combine objects with similar properties in one block of code with differences in methods.

7. Conclusion:

Overall, Polymorphism is essential in programming as it allows for creation of reusable code and the simplification of such in a block of code.

Please refer to this link:

https://github.com/PaulJustinePoletico/CPE-103-OOP-1-A/blob/main/Laboratory_Activity_No_7.ipynb

8. Assessment Rubric: