

Laboratory Activity No. 7

Polymorphism

Course Code: CPE103

Program: BSCPE

Course Title: Object-Oriented Programming

Date Performed: 02/22/25

Section: 1A

Date Submitted: 02/23/25

Name: Czer Justine D. Maringal

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:
```

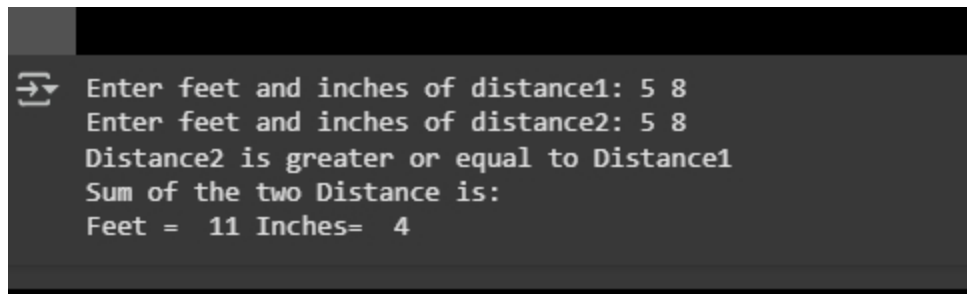
```
    print("Distance2 is greater or equal to Distance1")
```

```
d3=d1+d2
```

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

```
d3.show()
```

4. Screenshot of the program output:

A screenshot of a terminal window showing the output of a program. The text is as follows:

```
Enter feet and inches of distance1: 5 8
Enter feet and inches of distance2: 5 8
Distance2 is greater or equal to Distance1
Sum of the two Distance is:
Feet = 11 Inches= 4
```

Testing and Observing Polymorphism

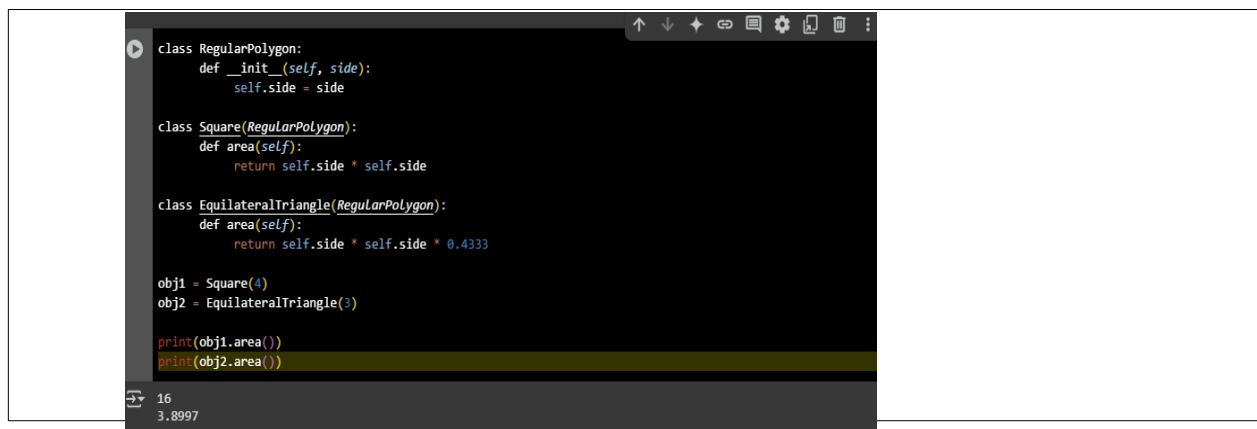
1. Create a code that displays the program below:

```
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 :

A screenshot of a Jupyter Notebook interface. The code cell contains the same Python code as in the previous block. The output cell shows the results of the area calculations:

```
16
3.8997
```

3. Run the program and observe the output.

4. **Observation:**

I observe that even if we use a class as many as we want it will carry all of its methods and will apply it to the It and always run even if we use multiple times as we see in the picture of the program.

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.

Proceed to this link: https://colab.research.google.com/github/ryaneulin/CPE-103-OOP-1-A/blob/main/Laboratory_Activity_No_7.ipynb

Questions

1. Why is Polymorphism important?

Polymorphism enhances code flexibility, reusability, and maintainability by allowing different classes to be treated as a common superclass. This simplifies function handling and reduces redundancy, making programs more scalable and efficient.

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

Polymorphism helps reduce code duplication, it can also introduce complexity and potential performance overhead due to dynamic method resolution, requiring careful implementation for efficiency.

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

CSV and JSON reading/writing offer flexibility by allowing efficient handling of multiple file formats. CSV is great for structured, tabular data, while JSON is ideal for storing hierarchical or complex data structures.

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

When implementing polymorphism, consider proper inheritance design, method overriding, performance trade-offs, and maintaining readability for scalability and efficiency.

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

It is commonly used in real world application like GUI frameworks where different UI elements repond to the same handler.

6. Conclusion:

Polymorphism allows different classes to be handled through a common interface, making code more flexible and reusable. It simplifies program design and supports scalability but requires careful implementation to manage complexity and performance trade-offs effectively.

8. Assessment Rubric: