

Laboratory Activity No. 7

Polymorphism

Course Code: CPE103

Program: BSCPE

Course Title: Object-Oriented Programming

Date Performed: 02/24/2025

Section: 1A

Date Submitted: 03/07/2025

Name: Ampong, j-kevin L.

Instructor: Maam 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:

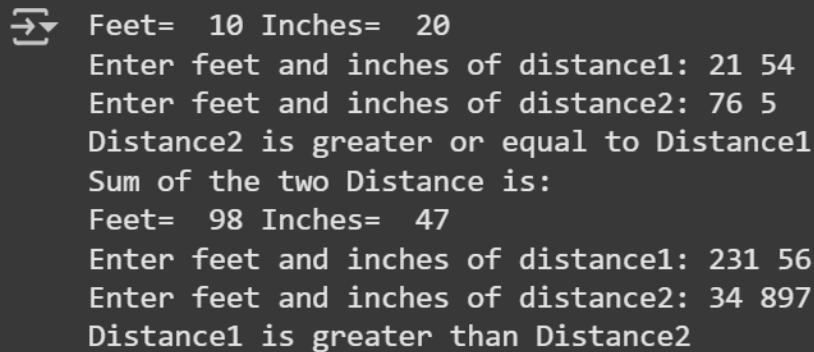
print("Distance2 is greater or equal to Distance1")

d3=d1+d2

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

d3.show()

4. Screenshot of the program output:



```
Feet= 10 Inches= 20
Enter feet and inches of distance1: 21 54
Enter feet and inches of distance2: 76 5
Distance2 is greater or equal to Distance1
Sum of the two Distance is:
Feet= 98 Inches= 47
Enter feet and inches of distance1: 231 56
Enter feet and inches of distance2: 34 897
Distance1 is greater than Distance2
```

Please refer to this link: <https://colab.research.google.com/drive/1ewD6R-7Vwh-D2FCYbphCsVVDYWg9cbhC#scrollTo=EZzG-d3qXHQn&line=2&uniqifier=1>

Testing and Observing Polymorphism

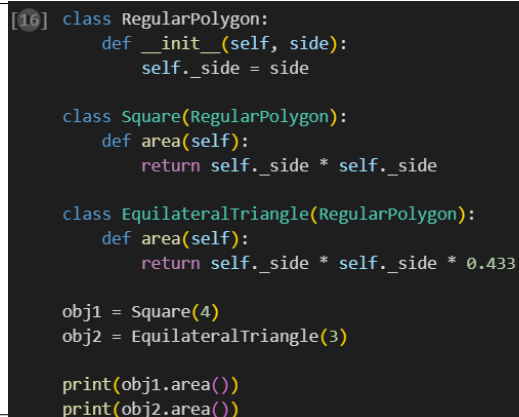
1. Create a code that displays the program below:

```
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:



```
[46] 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())
```

3. Run the program and observe the output.

Please refer to this link: <https://colab.research.google.com/drive/1ewD6R-7Vwh-D2FCYbphCsVVDYWg9cbhC#scrollTo=NOtymPw8ZL5v&line=7&uniqifier=1>

4. Observation: in reuse the `__init__` so it no longer needs to re-implement it. Another thing is that this applies the polymorphism since it represents both the square and equilateral triangle so that instead of rewriting the `__init__` on each child of the parent class, it just reuses it so there is no more duplication or repetition of it.

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.

Please refer to this link: <https://colab.research.google.com/drive/1ewD6R-7Vwh-D2FCYbphCsVVDYWg9cbhC#scrollTo=ztqbtQo4X8Jk>

Questions

1. Why is Polymorphism important?

Polymorphism is a part of OOP that offers the reusability of the code without removing the maintainability and scalability of the code since it's one of the 4 pillars of the OOP rendering, which is one of the essential foundations for creating the professional standard.

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

The main advantage of polymorphism is that it allows your code to be reused and modified without rewriting the whole source code of it. These properties of polymorphism can help us to avoid duplication and help us to write code that can be maintained for a long run. While there is an advantage of using it, there are also disadvantages to it, like it's only useful if a specific code needs to be used in many forms, but if you only need to use it once, then it becomes useless since polymorphism is only good if you have repeating code that needs to be reused more than once.

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

The biggest advantage is that it allows your data to be read and modified by another programming language and application, making it versatile when making transformable data since it's widely supported by many languages. Not only that, but it also makes storing data easy, allowing for portable data to be used. While there are many good things about it, we should consider that it takes a lot of time to get used to its data structure, but also bigger data on it may cause a memory problem since it's not the fastest option for transferring data, primarily to heavy applications.

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

f it's going to be reused later by another component of your program, you can avoid rewriting it the same way if it's particularly useful in that specific case.

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

Well for me based on information that I gather it use again to reduce rewriting the same program, and allows you program to be reuse, extends and modified making important component when writing real world application, there are probably many things that how it be used that I forget to mention but that's all I know currently but there many if may be use to tackle different problems

7. Conclusion:

Based on the answers we've gone through, it's clear that both polymorphism and working with different data formats like CSV and JSON are really important tools in programming, especially when we're thinking about building things in an object-oriented way. Polymorphism stands out as a key part of making our code reusable and easier to manage in the long run, which is super helpful for making programs that can grow and change without becoming a total mess. This idea of being able to reuse code in different forms is a big deal for making sure our programs are strong and built to a good, professional level.

But we also need to keep in mind that these tools aren't always the perfect answer for every situation. Polymorphism is great when we have code that we know we're going to use in lots of different ways, but maybe not so useful if we only need something once. And while CSV and JSON are fantastic for sharing data and making it work across different programs, they can have their downsides too, like needing to learn their structure or dealing with bigger files that can slow things down. So, thinking carefully about when and how to use these things is really important to make sure we're using them in the best way possible for what we're trying to build.

8. Assessment Rubric: