Laboratory Activity No. 7	
Polymorphism	
Course Code: CPE103	**Program:** BSCPE
Course Title: Object-Oriented Programming	**Date Performed:** Feb 30, 2025
Section: BS CpE 1A	**Date Submitted:** March 14, 2025
Name: Junichiro H. Uy	**Instructor:** Maria Rizette M. 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:

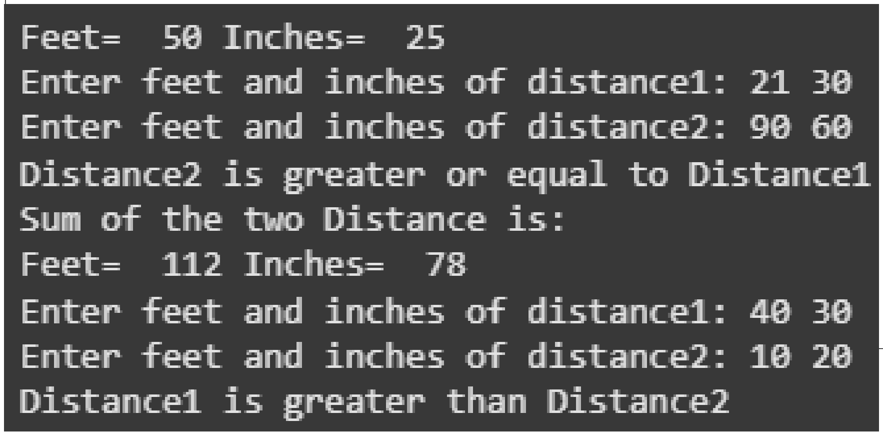
Creating the Classes

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:



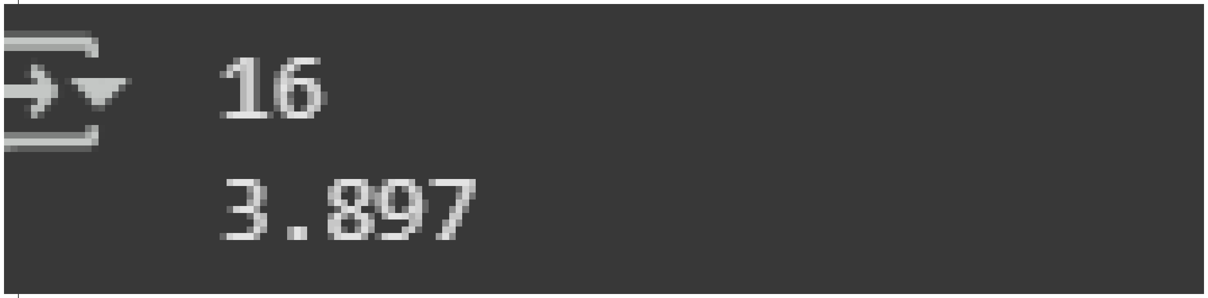
```
Feet=  50 Inches=  25
Enter feet and inches of distance1: 21 30
Enter feet and inches of distance2: 90 60
Distance2 is greater or equal to Distance1
Sum of the two Distance is:
Feet=  112 Inches=  78
Enter feet and inches of distance1: 40 30
Enter feet and inches of distance2: 10 20
Distance1 is greater than Distance2
```

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:

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.

Refer to this link:
https://colab.research.google.com/drive/1B8tombv0N04_zjIDVwGMIcu6j2bxbEFr#scrollTo=qsD4NfYEQ_Cf&line=48&uniqifier=1

Questions

1. Why is Polymorphism important?

- Polymorphism is like having a universal remote that can control different devices; it allows different classes to be treated as if they are the same type. This makes our code more flexible and easier to manage, as we can use the same functions for different objects.

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

- One of the great things about polymorphism is that it lets us reuse code, so we don't have to write the same thing over and over for different types of objects. It also makes it super easy to add new features or classes without messing up the existing code.
- On the flip side, polymorphism can make the code a bit more complicated, which might confuse someone who's just starting out. Plus, there can be a slight performance hit because the program has to figure out which method to call at runtime.

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

- Using CSV and JSON files is fantastic because they're widely accepted and easy for people to read and edit. However, CSV can struggle with complex data structures, and if the data isn't formatted correctly, it can lead to frustrating errors.

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

- When thinking about using polymorphism, it's important to make sure that all the classes involved share a common interface, kind of like a team working towards the same goal. Also, thorough testing is key to ensure everything behaves as expected when we use this flexibility.

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

- You can see polymorphism in action in user interfaces, where different elements like buttons and text fields can be handled in a similar way, making life easier for developers. It's also common in games, where various character types can share common actions, like moving or attacking, without needing separate code for each one.

7. Conclusion:

- Polymorphism is a powerful concept in object-oriented programming that enhances code flexibility and reusability by allowing different classes to be treated as the same type. While it simplifies code management, it can also introduce complexity and potential performance issues. Overall, when used thoughtfully, polymorphism can significantly improve the design and functionality of software applications.

8. Assessment Rubric: