Laboratory Activity No. 7	
Polymorphism	
Course Code: CPE103	**Program:** BSCPE
Course Title: Object-Oriented Programming	**Date Performed:** 02-22-25
Section: BSCpE – 1A	**Date Submitted:** 02-23-25
Name: Vasig, Yuan Hessed O.	**Instructor:** Engr. 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

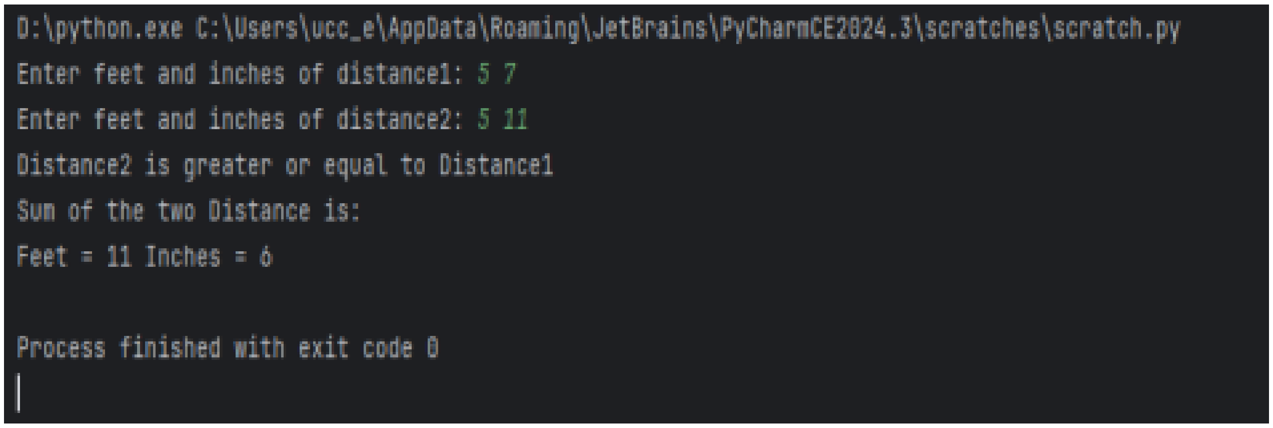
5. Procedure:

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()
```

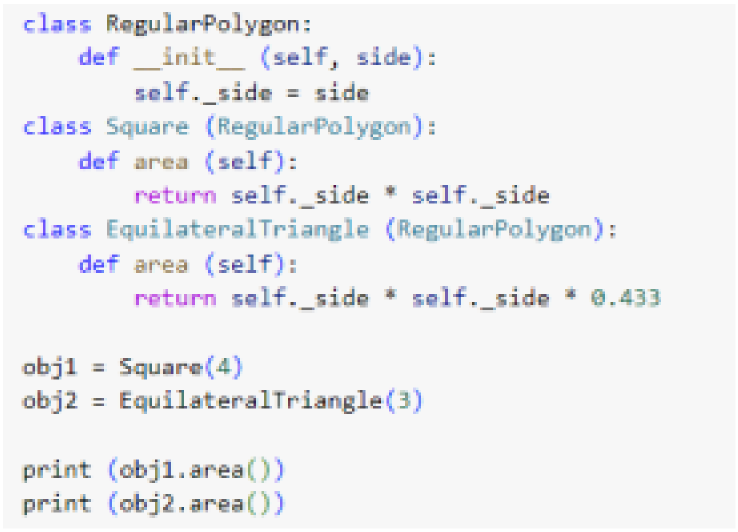
4. Screenshot of the program output:



```
D:\python.exe C:\Users\ucc_e\AppData\Roaming\JetBrains\PyCharmCE2024.3\scratches\scratch.py
Enter feet and inches of distance1: 5 7
Enter feet and inches of distance2: 5 11
Distance2 is greater or equal to Distance1
Sum of the two Distance is:
Feet = 11 Inches = 6

Process finished with exit code 0
```

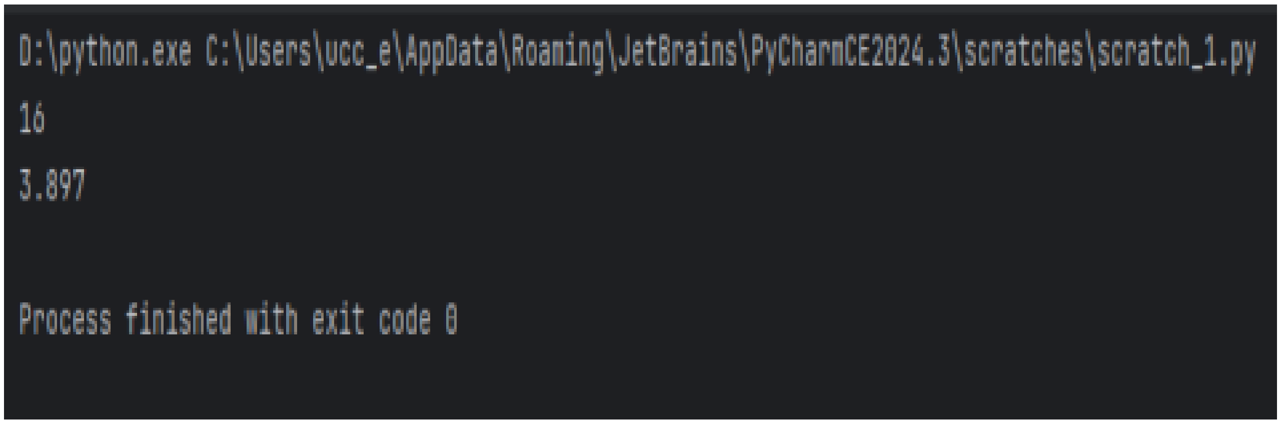
Testing and Observing Polymorphism

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:



```
D:\python.exe C:\Users\ucc_e\AppData\Roaming\JetBrains\PyCharmCE2024.3\scratches\scratch_1.py
16
3.897


Process finished with exit code 0
```

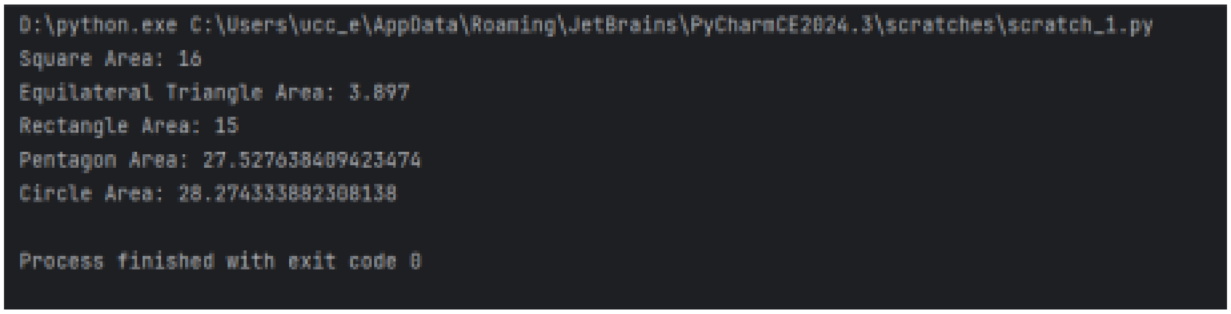
3. Run the program and observe the output.
4. Observation:

When polymorphism_a.py is executed, the program prompts the user for two distances in inches and feet. It then compares them and prints out which one is larger. Then it adds the two distances and prints out the sum. The program works properly when the inches are more than 12 by converting them into feet.

For polymorphism_b.py, the code creates two shapes: a square and an equilateral triangle. It computes and prints both shapes' area. The area of the square is computed by squaring its side, while the area of the triangle is computed using a formula that considers its shape. The output reflects the correct areas according to the provided side lengths.

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.



```
D:\python.exe C:\Users\ucc_e\AppData\Roaming\JetBrains\PyCharmCE2024.3\scratches\scratch_1.py
Square Area: 16
Equilateral Triangle Area: 3.897
Rectangle Area: 15
Pentagon Area: 27.527638409423474
Circle Area: 28.274333882308138

Process finished with exit code 0
```

In this program, the areas for five different shapes are calculated and displayed:

•The square which is a four-sided shape with equal lengths uses the formula side × side for determining its area, getting area=16 square units when length=4.
•The equilateral triangle with three equal sides uses the formula 0.433 × $side^2$ and gives an area of about 3.897 square units in case the side length is 3.
•The rectangle calculates the area using length × width and gives an area of 15 square units for a length of 5 and width of 3. Similarly, a special formula is used to calculate the area of the pentagon based on its regular shape: the derived formula becomes: (5 × $side^2$)/ (4 × tan(π/5)), thus leading to an area of almost 27.53 square units for a side length of 4.
•The circle employs the standard formula π × $radius^2$, which gives an area of 28.27 square units if the radius measures 3. Proper labeling makes every area related to its shape understandable in a value derivation sense, which you can do by taking a screenshot of output.

Questions

1. Why is Polymorphism important?

- Polymorphism is crucial in object-oriented programming (OOP) because it promotes code reusability, extensibility, and flexibility by allowing objects of different classes to be treated as objects of a common type, simplifying code and enabling easier maintenance and adaptation.

2. Explain the advantages and disadvantages of using applying Polymorphism in an Object-Oriented Program. - It lessens code reusability and flexibility and enhances the maintainability of the code because the same method allows us to operate on different objects. It aids understanding and ease of maintenance. The disadvantage of this is that it adds to the complexity of the system and reduces performance because dynamic method calls become much slower; this, in turn, makes debugging much harder. It may even lead to a greater use of memory. In the presence of all these shortcomings, polymorphism is nevertheless an asset when writing programs that require efficiency.

3. What maybe the advantage and disadvantage of the program we wrote to read and write csv and json files? - The program simplifies data storage and retrieval using CSV and JSON files. CSV is easy to read, while JSON is structured for web use. However, CSV lacks complex structure, and JSON can be hard to read. Errors in data and slow processing of large files are possible challenges. Still, it helps in efficient data management.

4. What maybe considered if Polymorphism is to be implemented in an Object-Oriented Program? - When implementing polymorphism in an object-oriented program, consider inheritance, interfaces, method overriding, and method overloading to allow objects of different types to be treated as objects of a common type, promoting code reusability and flexibility.

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

- Polymorphism in programming gives a program the ability to redefine methods for derived classes. Five different types of polymorphism that are commonly used in various programming languages are Ad-hoc, Parametric, Subtyping, Row, and Polytypism.

7. Conclusion:

In this lab, we investigated using Python to determine the area of various forms. We were able to determine the areas of a square, equilateral triangle, rectangle, pentagon, and circle by utilizing classes and formulas. The computer appropriately used the formulas for each shape to produce the desired outcomes. This exercise demonstrated how object-oriented programming facilitates calculation organization and simplification. It also showed how various shapes might use their own special formulas while sharing similar architecture. All things considered, this lab gave me a practical understanding of both programming and geometry.

8. Assessment Rubric: