Name: Hamsa M Ahmed
Subject: OOP Practical
Department: Computer Network System

Lecture 7: Polymorphism in Object-Oriented Programming

1. Introduction

Polymorphism means "many forms" and refers to the ability of different classes to be treated through a common interface. It allows methods to behave differently depending on the object calling them, which improves flexibility and maintainability.

2. Polymorphism with Method Overriding

When a child class provides a specific implementation of a method that is already defined in its parent class, it is called method overriding.

```
class Animal:
    def speak(self):
        print("Animal makes a sound")

class Dog(Animal):
    def speak(self):
        print("Woof!")

class Cat(Animal):
    def speak(self):
        print("Meow!")

animals = [Dog(), Cat(), Animal()]

for animal in animals:
    animal.speak()
```

Output:

```
Woof!
Meow!
Animal makes a sound
```

Name: Hamsa M Ahmed
Subject: OOP Practical
Department: Computer Network System

3. Polymorphism with Functions

Functions can accept objects of different classes that share the same method names and call these methods without knowing the exact class.

```
def make_animal_speak(animal):
    animal.speak()

dog = Dog()
cat = Cat()

make_animal_speak(dog)
make_animal_speak(cat)
```

4. Duck Typing in Python

Python follows "duck typing," meaning the suitability of an object is determined by the presence of certain methods and properties, not the actual type.

Example:

```
class Bird:
    def speak(self):
        print("Tweet")

class Airplane:
    def speak(self):
        print("Whoosh")

def make_speak(entity):
    entity.speak()

make_speak(Bird())
make_speak(Airplane())
```

5. Exercises

1. Create a base class `Shape` with method `area()`.
2. Create subclasses `Circle` and `Rectangle` that override `area()` with their specific calculations.
3. Write a function to accept a list of `Shape` objects and print their areas.

Name: Hamsa M Ahmed
Subject: OOP Practical
Department: Computer Network System

6. Summary

Polymorphism enables writing flexible and extendable code by allowing different object types to be used interchangeably based on shared behavior.

7. References

1. Python Docs - Polymorphism
2. Real Python - Polymorphism