Lecture slides - Week 11

OOP - Polymorphism

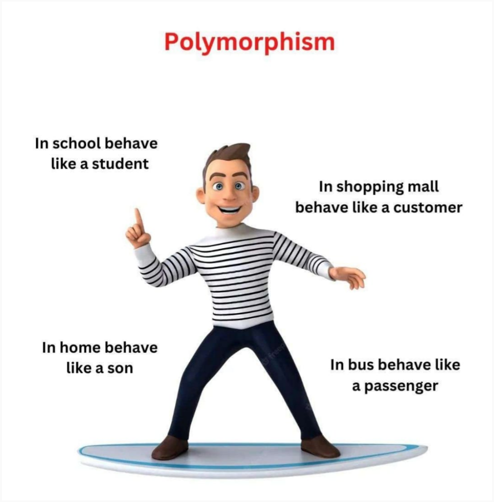
Dr. Aamir Akbar

Director of both AWKUM AI Lab and AWKUM Robotics, Final Year Projects (FYPs) coordinator, and lecturer at the department of Computer Science
Abdul Wali Khan University, Mardan (AWKUM)

Contents

Polymorphism

Polymorphism in real life and OOP i


Polymorphism
In school behave like a student
In shopping mall behave like a customer
In home behave like a son
In bus behave like a passenger

Polymorphism in real life and OOP ii

`Polymorphism` originates from the Greek words: `poly` meaning many and `morph` meaning form.

In OOP, `polymorphism` refers to the ability of different objects to be treated as instances of a common superclass, allowing them to exhibit different behaviors based on their specific types.

Two types of polymorphism:

1. `Compile-time Polymorphism or Static Binding` is achieved through `method overloading` which allows different functions or operators to be used with the same name within the same scope, and the compiler determines which function or operator to call based on the number and type of arguments. Note that Python is a `dynamically-typed` language, and it does not support traditional method overloading like some `statically-typed` languages (e.g., Java or C++), where you can have multiple methods with the same name but different parameter types or numbers.

Polymorphism in real life and OOP iii

2. Runtime Polymorphism or Dynamic Binding is achieved through method overriding, where a subclass provides a specific implementation of a method that is already defined in its superclass. The decision of which method to execute is made at runtime, based on the actual object type rather than the reference type.

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

class Dog(Animal):
    def sound(self):
        print("Dog barks")

class Cat(Animal):
    def sound(self):
        print("Cat meows")

animal = Dog()
animal.sound()  # Output: Dog barks

animal = Cat()
animal.sound()  # Output: Cat meows
```

The Object Class

What is the Object class in Python?

In Python, the `object` class is the base class for all other classes. It's at the top of the Python class hierarchy and provides default implementations for several special methods. All user-defined classes in Python implicitly inherit from the object class.

The `object` class mainly provides default behavior and attributes inherited by all other classes in Python.

Instance created directly of the `object` class typically don't contain much additional functionality unless you add attributes or methods to them within a subclass.

Case Study: Animal Battlefield

Animal Battlefield i

Welcome to the Animal Arena, a thrilling showdown where creatures from the wild come face to face in a battle of strength and strategy!

Image source

Animal Battlefield ii

In this arena, we'll be simulating epic battles between fierce animals — **a Lion and an Elephant**.

Image source

Activity: create a Python program that models these animals' attributes such as `name and health` and orchestrates their battles behaviors such as `attack and defense` using OOP principles: Inheritance and Polymorphism.

Modeling attack and defense behaviors i

The choice of attack and defense values for the Lion and Elephant classes can reflect the perceived characteristics and behaviors of these animals in a battle scenario.

Lion:

- Attack (5 to 15): Lions are known for their agility, strength, and powerful strikes when hunting or in confrontations. The attack range of 5 to 15 signifies the lion's forceful attacks, potentially causing significant damage.

- Defense (1 to 5): Lions might not have a high defensive capability as their offensive strength. The lower defense range imply a slightly lower capacity to withstand direct hits.

Modeling attack and defense behaviors ii

Elephant:

- Attack (3 to 10): Elephants are incredibly strong, however, their attack range might be set slightly lower than the lion's to show their strength while accounting for their potentially slower or less agile strikes.

- Defense (3 to 8): Elephants, while strong, might have a better ability to defend themselves due to their size and thick skin. The defense range of 3 to 8 signifies their capacity to absorb damage or fend off attacks more effectively compared to their attack potential.

Game Logic

A Battle class that orchestrates the showdown between these animals. Simulate each turn where one animal attacks the other randomly, deducting health based on the attack and defense strategies. Continue this until one of the animals' health drops to zero or below.

The battle should display each attack, defense, and damage dealt. Include messages for successful attacks, blocked attacks, and when an animal gets defeated.