Real-World Example: Zoo Animal Hierarchy

Consider a zoo management system where animals are classified into different categories based on their characteristics. We can create an inheritance hierarchy to represent this classification:

Base Class: Animal

Common properties: name, age, sound.

Common methods: makeSound(), eat().

Derived Class 1: Mammal

Inherits from Animal.

Additional properties: furColor, liveBirth.

Additional methods: nurseYoung().

Derived Class 2: Bird

Inherits from Animal.

Additional properties: wingSpan, canFly.

Additional methods: buildNest().

Is-A Relationships:

Every Mammal Is-A Animal.

Every Bird Is-A Animal.

Instances:

Instance 1: Lion

Object of the Mammal class.

Inherits properties and methods from the Animal class.

Has additional properties like furColor.

Can call methods like makeSound() and nurseYoung().

Instance 2: Eagle

Object of the Bird class.

Inherits properties and methods from the Animal class.

Has additional properties like wingSpan.

Can call methods like makeSound() and buildNest().

Explanation:

In this example, the base class **Animal** provides a generic representation of an animal with common properties and behaviors. The derived classes **Mammal** and **Bird** inherit from **Animal** and add specific properties and methods that are characteristic of mammals and birds, respectively.

The instances, such as a Lion and an Eagle, are objects of the derived classes. They inherit the common properties and methods from the base class and have additional features specific to their categories.

This hierarchy allows for a clear and organized representation of zoo animals, capturing the commonality and specialization among different types of animals. It aligns with the real-world concept that mammals and birds are two distinct categories of animals within the broader classification of all animals.