****

**OOP Relationship Explanation Through Assignment Task 1**

1. **Inheritance**: Inheritance is when a class (child/subclass) derives properties and behaviors from another class (parent/superclass). In the given code, **Lion**, **Elephant**, and **Monkey** classes are inheriting from the **Animal** class. This means they get the **Name**, **Age**, **Species**, **Eat()**, etc. from the **Animal** class and can also have their own unique properties and methods.
2. **Association**: Association is a relationship where all objects have their own lifecycle and there is no owner. The objects can remain in contact with each other for a certain period of time. In the code, the **Zoo** class uses **Animal** and **Habitat** objects in its methods. The **Zoo** doesn't own these objects, but it interacts with them, creating an association.
3. **Aggregation**: Aggregation is a specialized form of Association where all objects have their own lifecycle, but there is an ownership, and child objects cannot belong to another parent object. In the code, **Zoo** class has a list of **Habitat** objects, this forms an aggregation. The **Zoo** owns **Habitats**, but if a **Zoo** is deleted, the **Habitat** can exist independently.
4. **Composition**: Composition is a specialized form of Aggregation where the child object does not have their lifecycle and if parent object deletes, all child objects will also be deleted. In the code, the **Habitat** class has a list of **Animal** objects and manages their lifecycle. If a **Habitat** object is deleted, all **Animal** objects within that **Habitat** would be logically deleted as well, hence, it's a composition.
5. **Dependency**: Dependency is a relationship where a class depends on another class, and if the latter class changes, it might affect the former class. The **Zoo** class depends on the **Animal** and **Habitat** classes. If either of these classes changes, it could affect the functionality of the **Zoo** class.
6. **Realization/Implementation**: Realization is a relationship between the interface and the class that implements the interface. A class that implements an interface must provide an implementation for all the methods declared in the interface. In the code, **Lion**, **Elephant**, and **Monkey** classes implement the **ISoundBehaviour** interface and hence they need to provide implementation for the **MakeSound()** method.