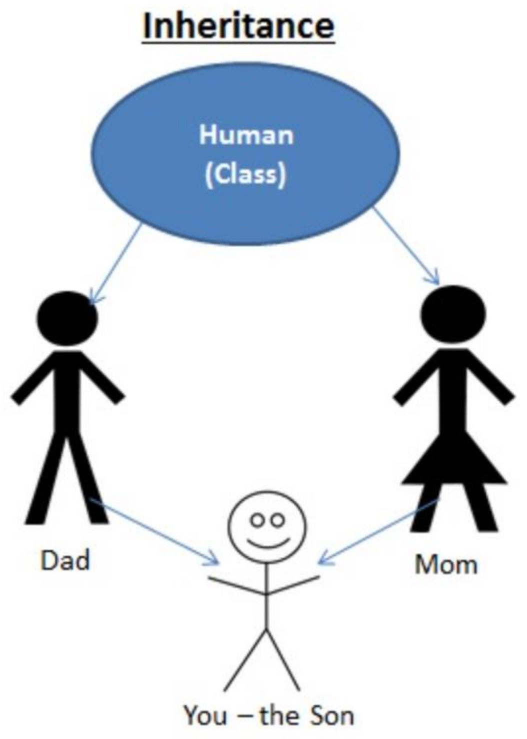
IT LEGEND

Create legends Programming

Inhertance and polmerphism in C#

| www.itlegend.net | 01013855500 | Ceo@itlegend.net

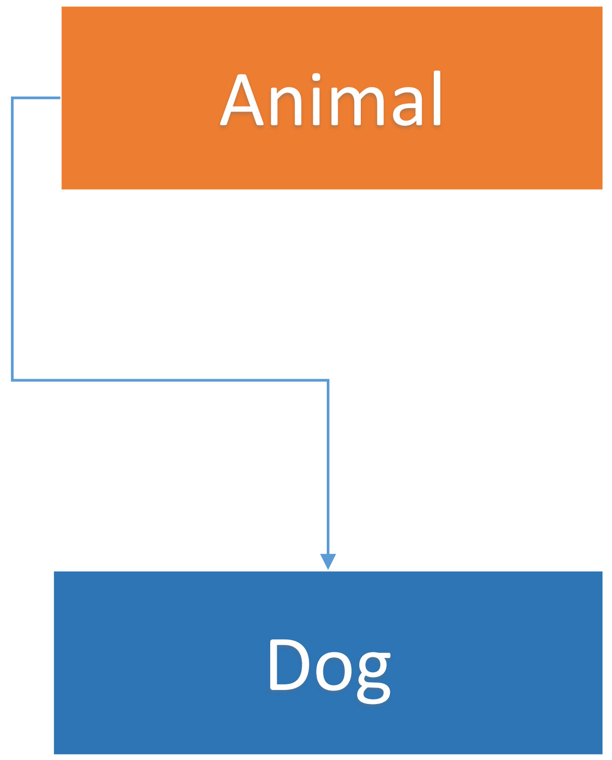


Inheritance

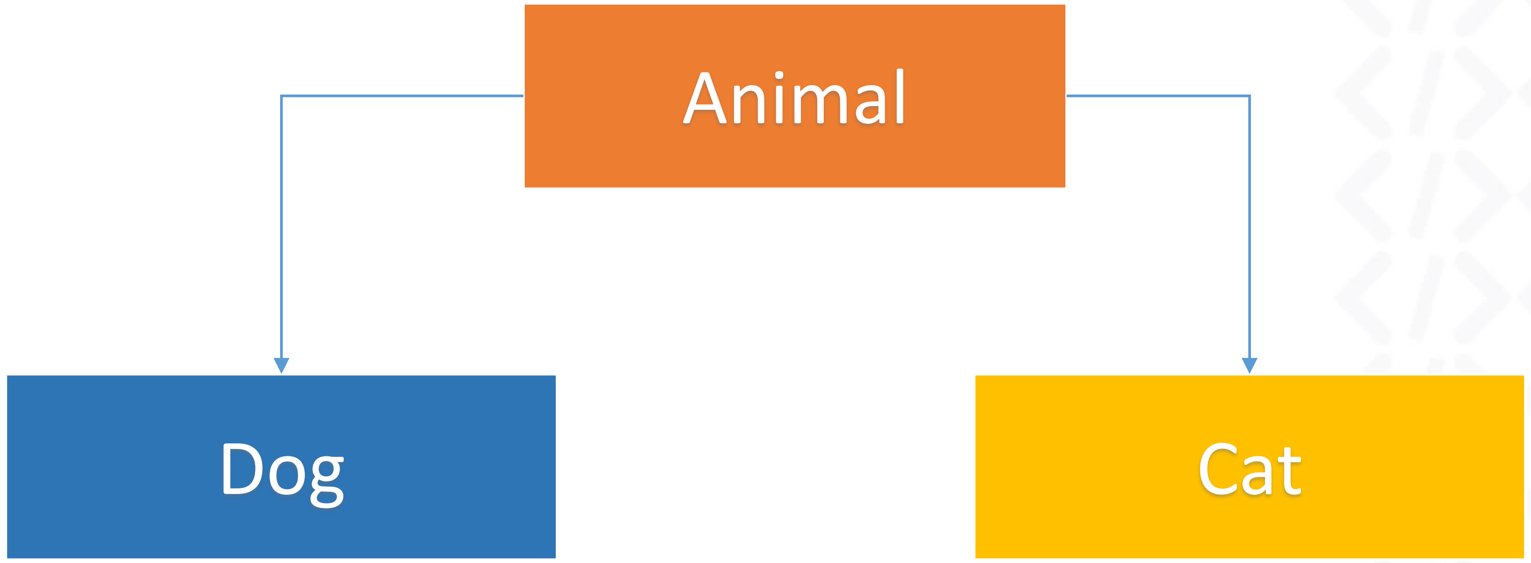
Inheritance

- Inheritance is a fundamental concept in object-oriented programming that allows us to define a new class based on an existing class. The new class inherits the properties and methods of the existing class and can also add new properties and methods of its own. Inheritance promotes code reuse, simplifies code maintenance, and improves code organization.

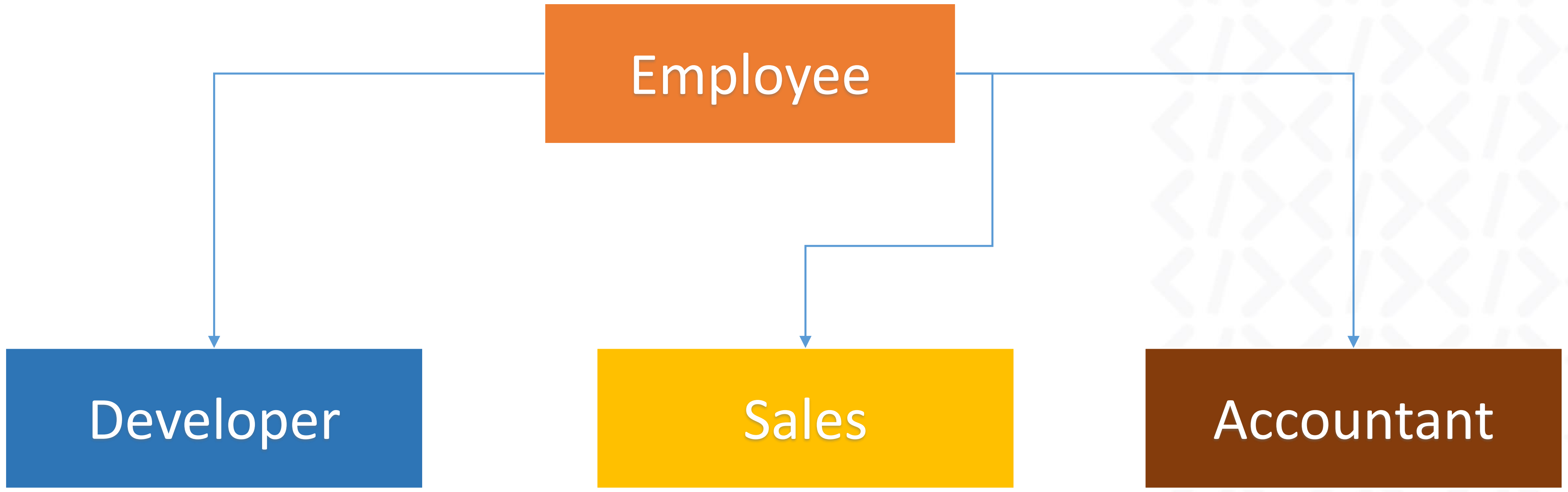
Single Inheritance



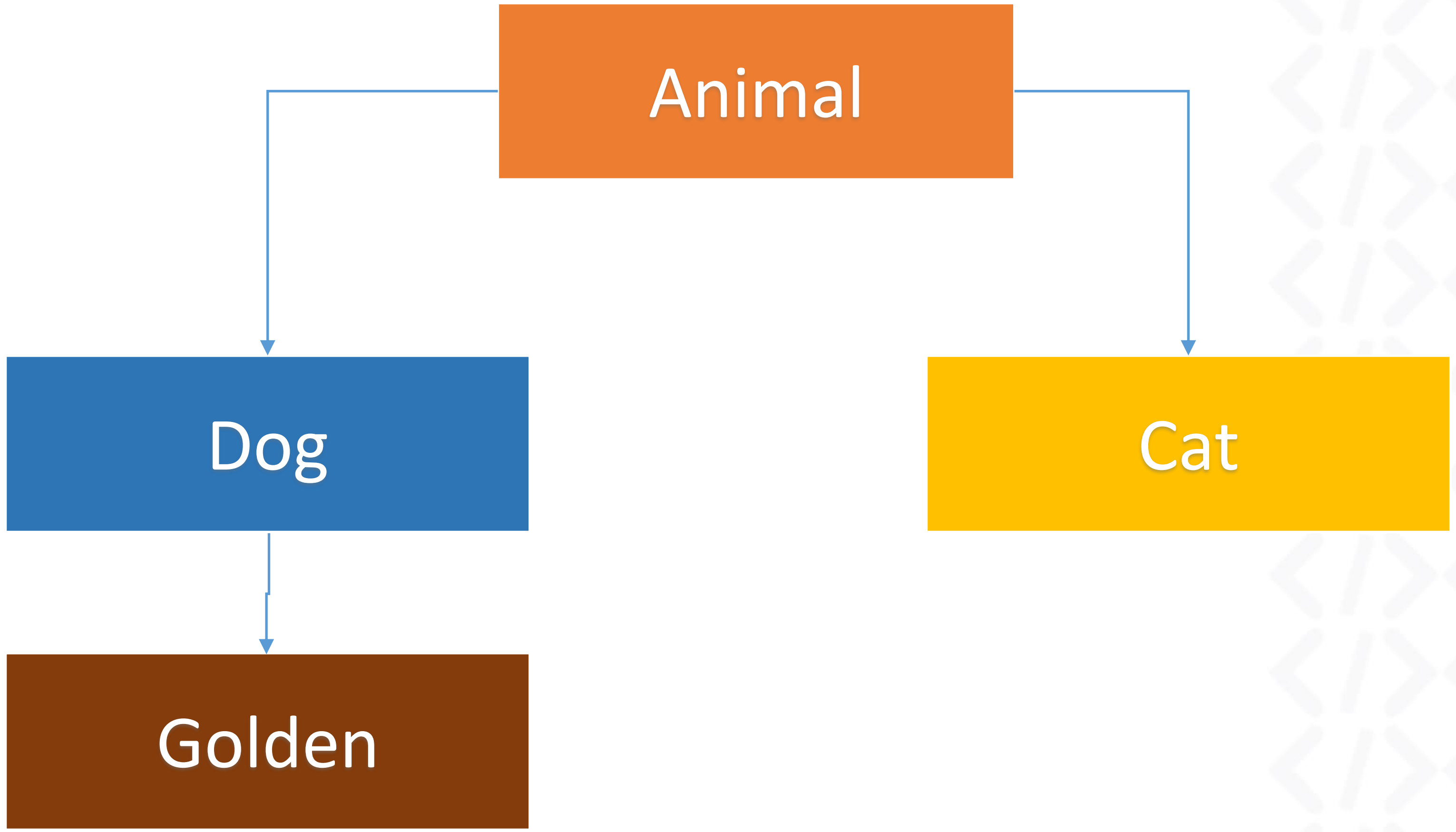
Hierarchical Inheritance



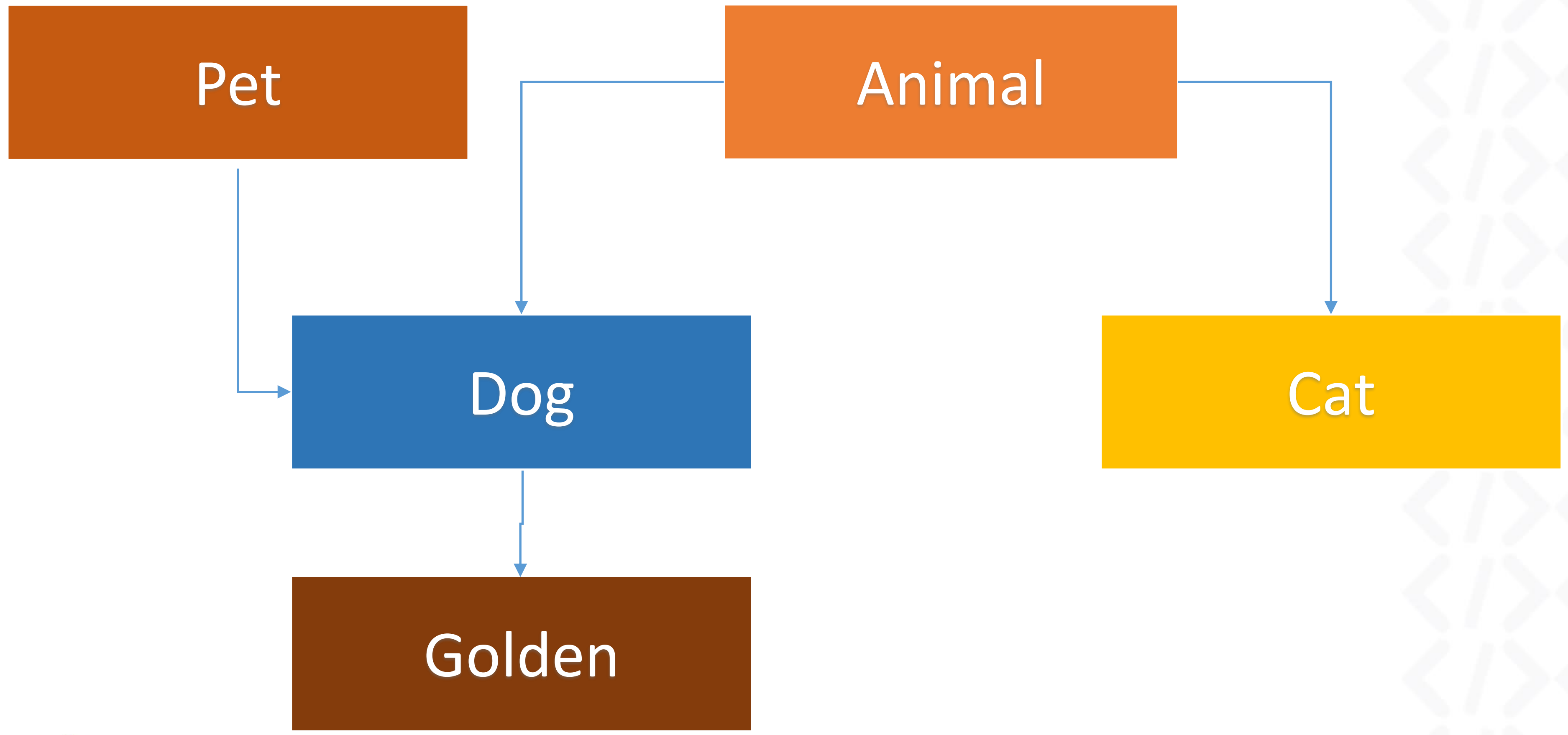
Hierarchical Inheritance



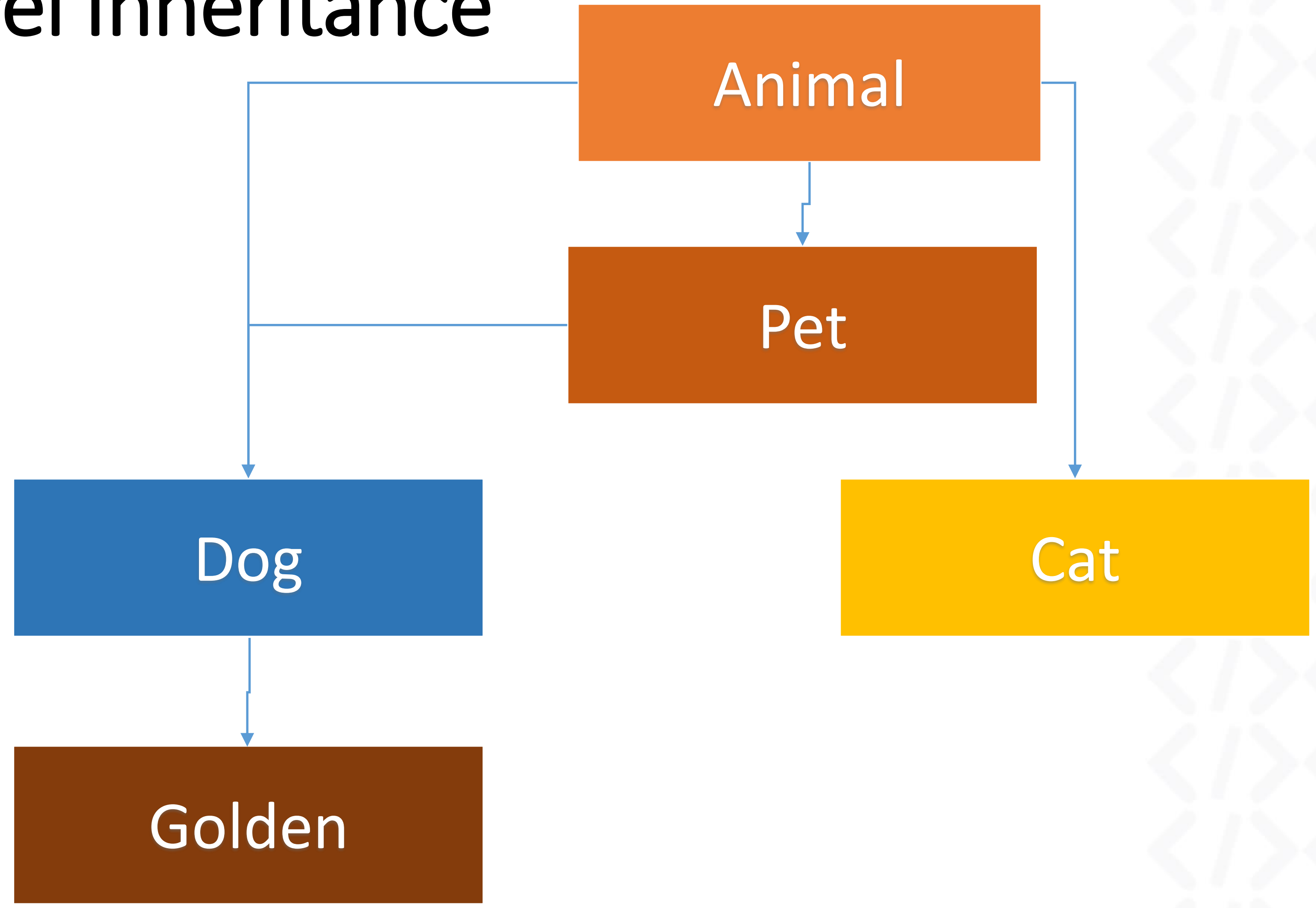
Multilevel Inheritance



Multiple Inheritance



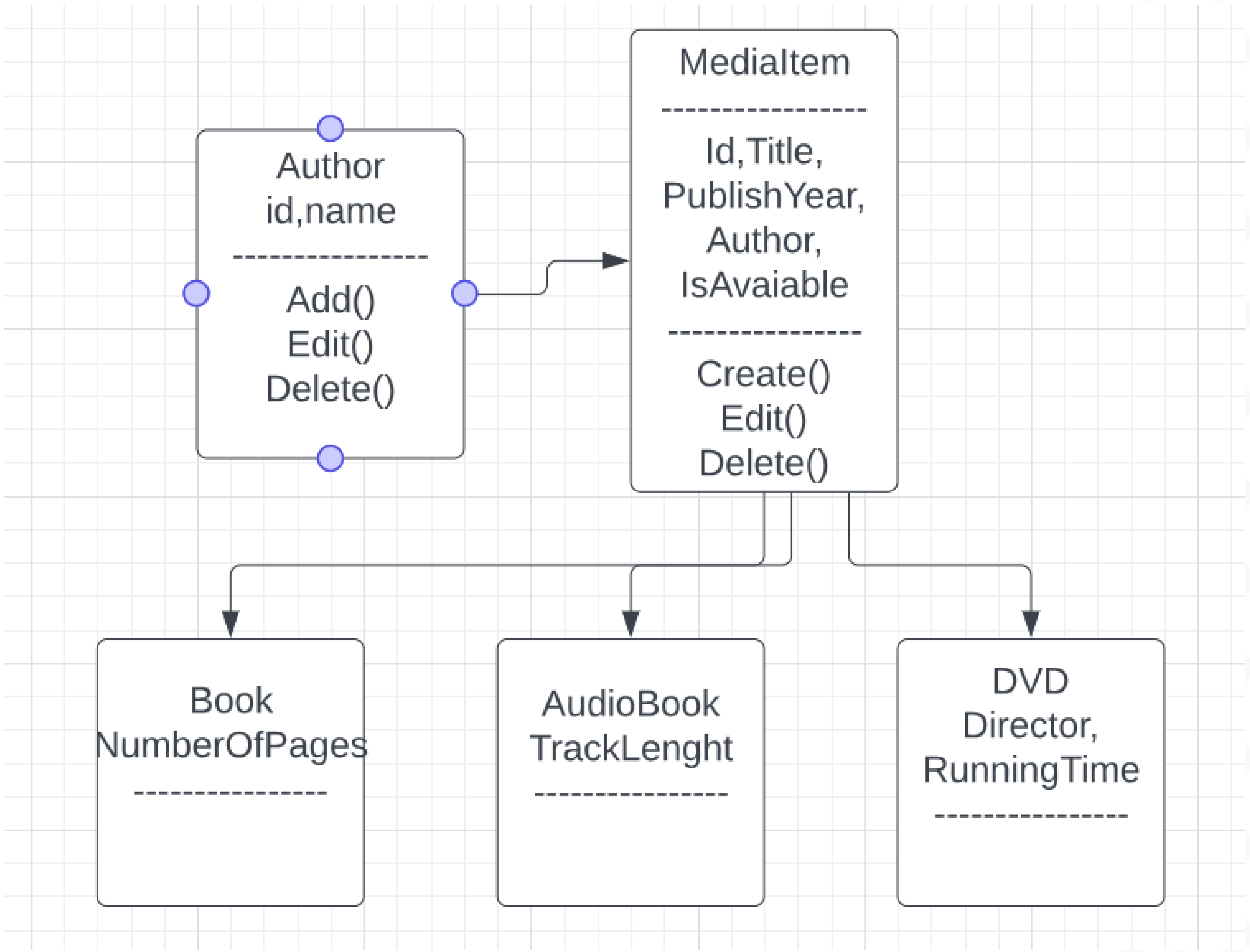
Multilevel Inheritance



Important points

- First class called super class or base class
- Derived class called subclass
- Superclass can only be one: A superclass can have any number of subclasses. But a subclass can have only one superclass. This is because C# does not support multiple inheritance with classes. Although with interfaces, multiple inheritance is supported by C#.
- Class can not inherit private fields

Example


Author
id,name

Add()
Edit()
Delete()
MediaItem

Id,Title,
PublishYear,
Author,
IsAvaiable

Create()
Edit()
Delete()
Book
NumberOfPages

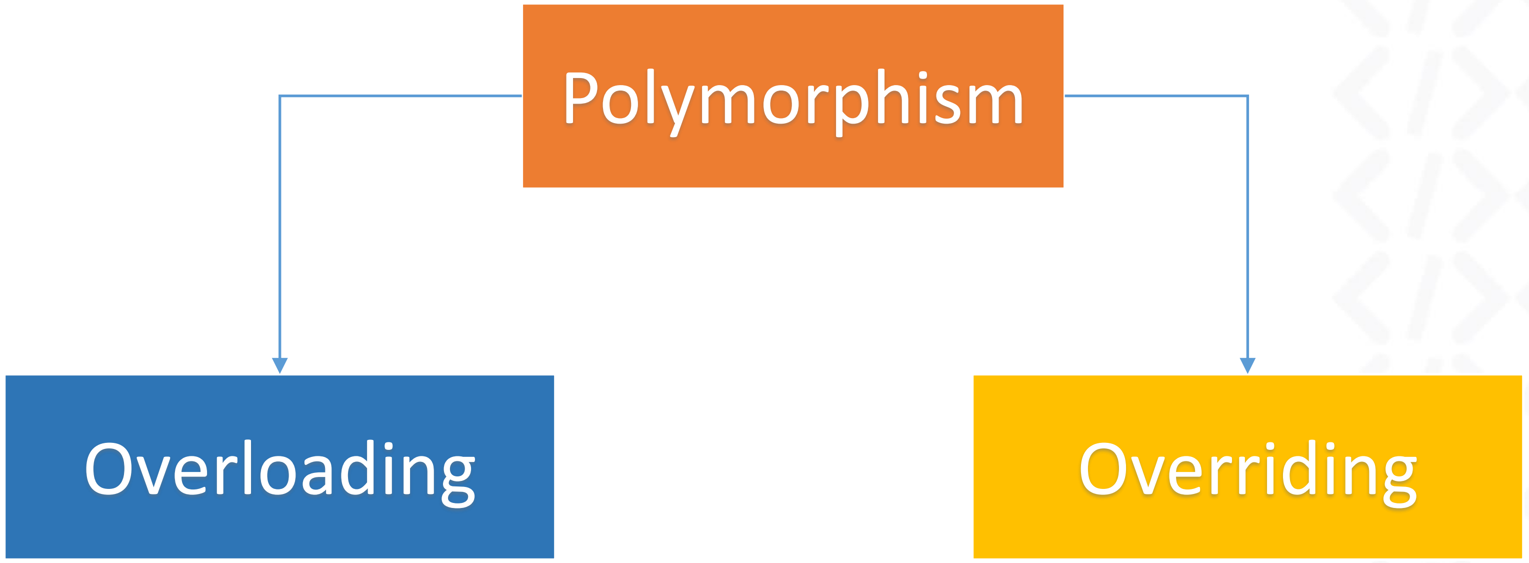
AudioBook
TrackLenght

DVD
Director,
RunningTime

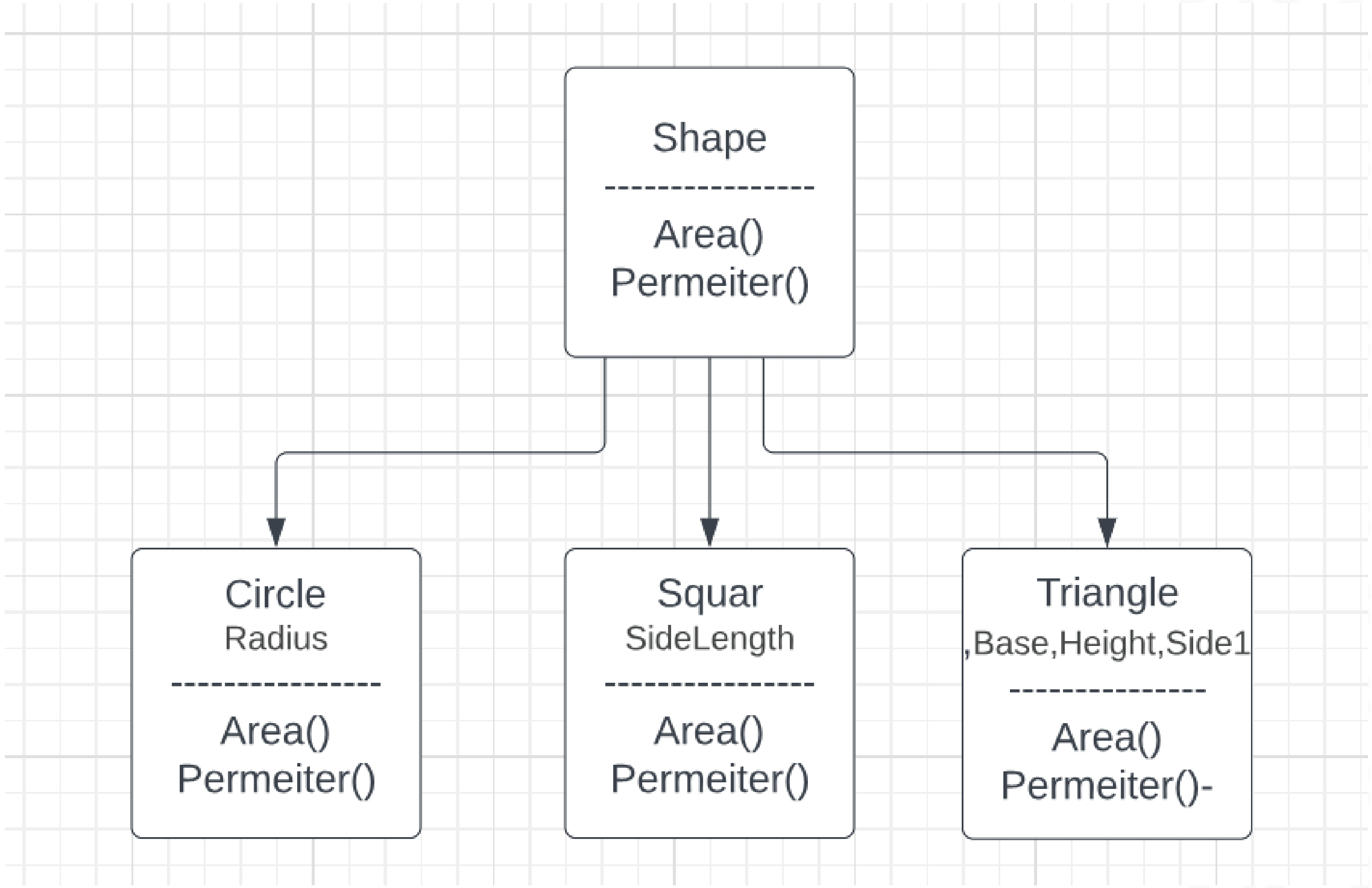
Polymorphism

- Polymorphism is a fundamental concept in object-oriented programming (OOP) that allows objects of different classes to respond to the same method call in different ways. It promotes code flexibility, reduces redundancy, and enables writing more generic code.

Polymorphism



Example


Shape

Area()
Permeiter()
Circle
Radius

Area()
Permeiter()
Squar
SideLength

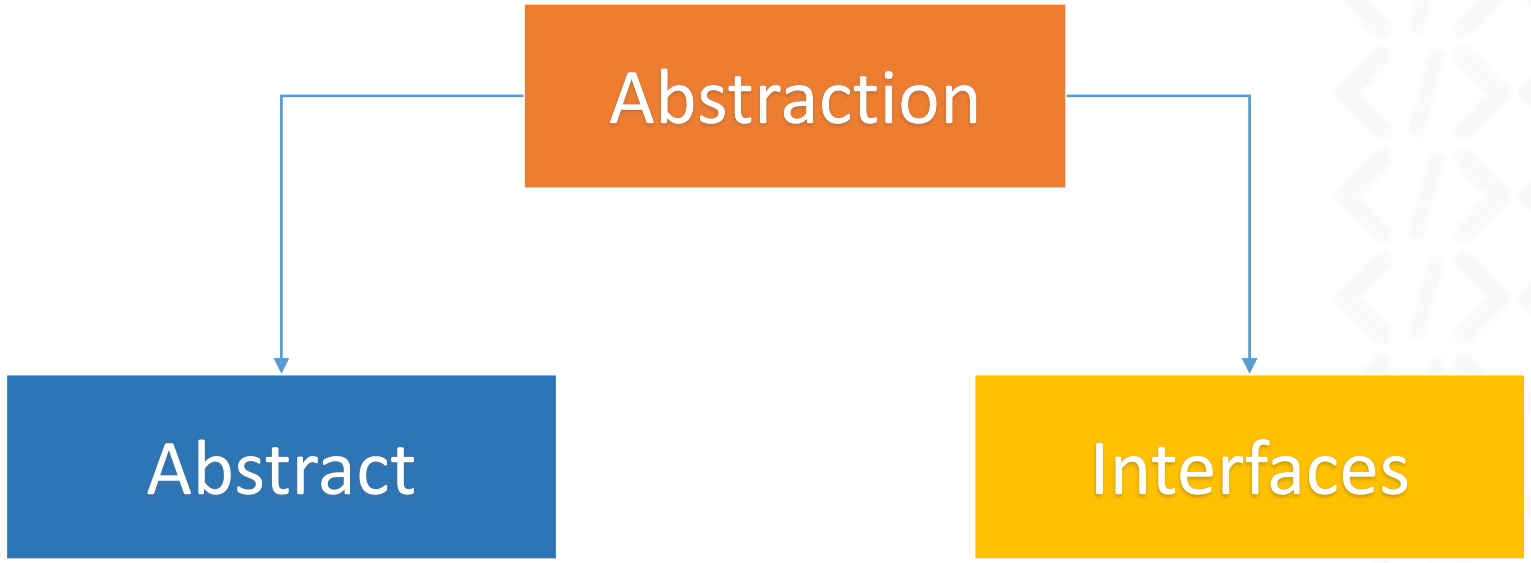
Area()
Permeiter()
Triangle
,Base,Height,Side1

Area()
Permeiter()-

Abstraction

- Abstraction is a fundamental concept in OOP that allows you to focus on the "what" (functionality) of an object rather than the "how" (implementation details). It promotes code reusability, maintainability, and reduces complexity by hiding unnecessary details.

Abstraction



Abstraction vs Polymorphism

Feature	Abstraction	Polymorphism
Concept	Focuses on hiding implementation details and exposing essential functionalities.	Ability of objects of different classes to respond differently to the same method call.
Mechanism	Achieved through abstract classes, interfaces.	Achieved through method overriding (virtual, abstract, and override keywords) and operator overloading.
Purpose	Simplifies code, promotes reusability, and improves maintainability by focusing on "what" objects do rather than "how" they do it.	Enables writing generic code that can work with various object types without knowing their specific implementations.
Example	Shape abstract class with abstract method Draw(). Derived classes like Circle and Square implement Draw() with their specific drawing logic.	List<T> collection class can hold objects of different types (e.g., int, string), and its methods like Add() work polymorphically.

Abstract class vs class vs Interface

Feature	Abstract Class	Class	Interface
Purpose	Provides a blueprint for derived classes, allowing partial abstraction and inheritance.	Serves as the basic building block for objects, encapsulating data (fields) and behavior (methods).	Defines a contract for functionality that implementing classes must provide, promoting loose coupling.
Instantiation	Cannot be directly instantiated.	Can be directly instantiated to create objects.	Cannot be directly instantiated.
Members	Can contain both abstract and non-abstract (concrete) methods, fields, properties, constructors.	Can contain only non-abstract methods, fields, properties, constructors.	Can only contain abstract methods (no implementation), static and final constants.
Inheritance	Can be inherited by one derived class.	Can inherit from another class (single inheritance).	Can be implemented by multiple classes (multiple inheritance).
Abstraction Level	Provides partial abstraction (can have some concrete methods).	No abstraction (all methods are concrete).	Provides full abstraction (all methods are abstract).