Inheritance in OOP

Inheritance is a core principle in OOP that allows a new class, called a subclass or derived class, to inherit properties and behaviors from an existing class, known as a superclass or base class. This fosters code reuse, promotes a hierarchical structure, and enables the creation of specialized classes based on more general ones. Inheritance models the "is-a" relationship, signifying that a subclass is a specialized version of its superclass.

Example

Let's consider a scenario involving different types of animals. We can create a base class called `Animal` that captures common properties and behaviors shared by all animals. Subsequently, we can derive specific classes like `Bird` and `Mammal` that inherit from the `Animal` class, adding specialized features.

```
class Animal {
  protected name: string;
  protected sound: string;

  constructor(name: string, sound: string) {
    this.name = name;
    this.sound = sound;
  }

  makeSound(): void {
    console.log(`${this.name} says ${this.sound}.`);
  }
}

class Bird extends Animal {
  private wingspan: number;

  constructor(name: string, sound: string, wingspan: number) {
    super(name, sound);
    this.wingspan = wingspan;
  }

  fly(): void {
    console.log(`${this.name} is flying
    with a wingspan of ${this.wingspan} meters.`);
  }
}

class Mammal extends Animal {
  private furColor: string;

  constructor(name: string, sound: string, furColor: string) {
    super(name, sound);
    this.furColor = furColor;
  }

  displayFurColor(): void {
    console.log(`${this.name}'s fur color
    is ${this.furColor}.`);
  }
}
```

In this example, the `Bird` ;and `Mammal` classes inherit from the `Animal` class. They not only share the common properties and method (`name`, `sound`, and `makeSound()`) but also introduce their own unique characteristics (`wingspan` and `fly()`for `Bird`, and `furColor` and `displayFurColor()` for `Mammal`). Through inheritance, we've created a hierarchy that promotes code reuse, abstraction, and extensibility. This enables the development of more specialized classes while maintaining a consistent structure. Understanding inheritance is essential for beginners as it forms the basis for creating modular and scalable object-oriented systems.