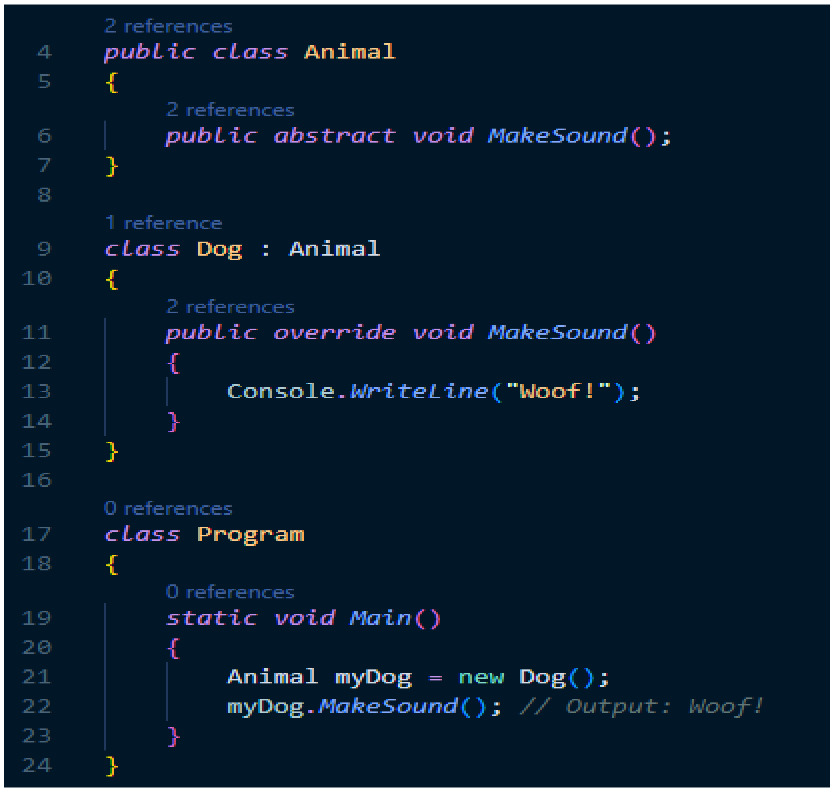
Abstraction is a fundamental concept in object-oriented programming (OOP) that helps simplify complex systems by hiding unnecessary details and exposing only what is essential. This makes code easier to understand, maintain, and reuse.

One major benefit of abstraction is that it reduces complexity. Instead of worrying about how something works internally, we can focus on using it. For example, when driving a car, we don't need to know how the engine works; we just use the pedals and steering wheel.

In programming, abstraction is commonly applied through classes and methods. Consider this Python example:



```
2 references
public class Animal
{
    2 references
    public abstract void MakeSound();
}

1 reference
class Dog : Animal
{
    2 references
    public override void MakeSound()
    {
        Console.WriteLine("Woof!");
    }
}

0 references
class Program
{
    0 references
    static void Main()
    {
        Animal myDog = new Dog();
        myDog.MakeSound(); // Output: Woof!
    }
}
```

Here, the Animal class defines an abstract method make_sound(), but does not implement it. The Dog class provides the specific implementation. This ensures that all animals must define make_sound(), while keeping the details flexible.

Abstraction improves code structure, making programs more scalable and easier to modify.