Inheritance and Polymorphism in C#

Step 1: preparing the Application

You’ll create a small application using the Visual Studio Code console application you created at the start of the course. Remove any existing code in the Program.cs file of your console application and create all the code in each step in that file.

Step 2: creating a Base Class and Derived Classes

Create a base class called **Animal** and two derived classes **Dog** and **Cat**. This introduces you to the concept of inheritance, where **Dog** and **Cat** inherit properties and methods from **Animal**.

Instructions:

1. Define a base class **Animal** with a **virtual** method **MakeSound**;
2. Create two derived classes **Dog** and **Cat** that inherit from **Animal**;
3. Override the **MakeSound** method in each derived class.

Step 3: using the virtual and override Keywords

Explore how to call methods in a base class you created using the virtual and override keywords.

Instructions:

1. Above any existing classes in Program.cs, create a Program class;
2. In the Program class, create a Main method;
3. In the Main method, create instances of the Dog and Cat classes and then call the MakeSound method from instances of Dog and Cat.
4. To check your answer, run the Visual Studio Code console application. If you receive an error when you run the code, go to the reading on the next page to compare your code to the correct answer.

Step 4: implementing Interfaces

Introduce interfaces to define a contract that classes can implement. Interfaces allow us to specify a set of methods that different classes must have.

Instructions:

1. Above any existing classes in Program.cs, define an interface called IAnimal with a method Eat;

2. Implement this interface in the Animal class and provide an implementation in the Dog and Cat classes.

Step 5: using the Interface

Explore how to call the interface methods from your main program.

Instructions:

1. In the Main method, underneath the existing method calls, call the Eat method from instances of Dog and Cat;
2. To check your answer, run the Visual Studio Code console application. If you receive an error when you run the code, go to the reading on the next page to compare your code to the correct answer.

Step 6: polymorphism with Lists of Base Types and Interfaces

Use polymorphism to interact with objects of different classes through a common base type or interface. This allows us to call methods on different objects in a unified way.

Instructions:

1. Update the Main method by creating a list of Animal objects that includes instances of Dog and Cat;
2. Use a loop to call the MakeSound method on each object in the list;
3. To check your answer, run the Visual Studio Code console application. If you receive an error when you run the code, go to the reading on the next page to compare your code to the correct answer

Code:

```
// Step 4.1: Define an interface IAnimal with a method Eat
interface IAnimal
{
    void Eat();
}

// Step 2.1 + Step 4.2: Base class Animal implements IAnimal
class Animal : IAnimal
{
    public virtual void MakeSound()
    {
        Console.WriteLine("The animal makes a sound.");
    }

    public virtual void Eat()
    {
        Console.WriteLine("The animal eats food.");
    }
}

// Step 2.2 + 2.3 + Step 4.2: Derived class Dog
class Dog : Animal
{
    public override void MakeSound()
    {
        Console.WriteLine("The dog barks: Woof!");
    }

    public override void Eat()
    {
        Console.WriteLine("The dog eats bones.");
    }
}

// Step 2.2 + 2.3 + Step 4.2: Derived class Cat
class Cat : Animal
{
    public override void MakeSound()
    {
        Console.WriteLine("The cat meows: Meow!");
    }

    public override void Eat()
    {
        Console.WriteLine("The cat eats fish.");
    }
}

// Step 3 + Step 5 + Step 6: Program class
class Program
{
    static void Main(string[] args)
    {
        // Step 3: Create Dog and Cat instances and call MakeSound
        Dog myDog = new();
        Cat myCat = new();

        myDog.MakeSound(); // The dog barks: Woof!
        myCat.MakeSound(); // The cat meows: Meow!

        // Step 5.1: Call Eat method from Dog and Cat instances
        myDog.Eat();       // The dog eats bones.
        myCat.Eat();       // The cat eats fish.

        // Step 6.1: Create a list of Animal objects
        List<Animal> animals =
        [
            new Dog(),
            new Cat()
        ];

        // Step 6.2: Use a loop to call MakeSound
        Console.WriteLine("\nPolymorphic behavior using base class
reference:");
        foreach (Animal animal in animals)
        {
            animal.MakeSound();
        }
    }
}
```