**АБСТРАКТНЫЙ КЛАСС (БАЗОВЫЙ) БАЗОВОЕ ЖИВОТНОЕ**

**abstract class BaseAnimal**

{

abstract public void DoSomeThing(); //ВСЕ ЖИВОТНЫЕ ЧТО ТО ДЕЛАЮТ

}

**ПРОИЗВОДНЫЙ ОТ АБСТРАКТНОГО и ОТ ИНТЕРФЕЙСА ЖИВОТНОЕ**

**class Animal: BaseAnimal,IVoice**

{

protected string \_name; //КЛИЧКА ЖИВОТНОГО

public Animal():this("Noname") { }

public Animal(string name)

{

\_name = name;

}

public override void DoSomeThing() //ПЕРЕОПРЕДЕЛЕНИЕ метода абстр класса

{

Console.WriteLine($"{**GetType().Name} {\_name} что-то делает**");

}

public void Eat() //ЖИВОТНОЕ ЕСТ (обычный метод)

{

Console.WriteLine($"{GetType().Name} {\_name} ест");

}

public virtual void Sleep() //ЖИВОТНОЕ СПИТ(виртуальный метод)

{

Console.WriteLine($"{GetType().Name} {\_name} спит");

}

public void Voice() //ЖИВОТНОЕ ИМЕЕТ ГОЛОС(обычный метод)

{

Console.WriteLine($"{GetType().Name} {\_name} имеет голос");

}

}

**ПРОИЗВОДНЫЙ ОТ ОБЫЧНОГО КЛАССА и ИНТЕРФЕЙСА КОРОВА ЖИВОТНОЕ**

**class Cow:Animal,IVoice**

{

int \_milk; //СОБСТВ ДАННОЕ МОЛОКО

public Cow() : base()

{

\_milk = 10;

}

public Cow(string name, int milk) : base(name)

{

\_milk = milk;

}

public new void Eat() //Обычный метод ЕСТ -НОВАЯ РЕАЛИЗАЦИЯ

{

Console.WriteLine($"{GetType().Name} {\_name} ест траву и сено");

}

public override void Sleep() ///Переопределение виртуального метода СПИТ

{

Console.WriteLine($"{GetType().Name} {\_name} спит стоя");

}

//сокрытие базовой реализации

public new void Voice() //Обычный метод ГОЛОС -НОВАЯ РЕАЛИЗАЦИЯ

{

Console.WriteLine($"{GetType().Name} {\_name} мычит");

}

void IVoice.Voice() // **ЯВНАЯ!!** РЕАЛИЗАЦИЯ МЕТОДА ГОЛОС У ИНТЕРФЕЙСА!

{

Console.WriteLine($"{GetType().Name} {\_name}: МУ-МУ");

}

}

**ОПРЕДЕЛЕНИЕ ИНТЕРФЕЙСА**

interface IVoice

{

void Voice();

}

**ОСНОВНАЯ ПРОГРАММА**

Console.OutputEncoding = System.Text.Encoding.Default;

**Animal cowA = new Cow("CowA",12); //создаем животное как корову**

**Cow cow = new Cow("Cow",5); //создаем корову как корову**

Console.WriteLine("--------EAT----------");

//невиртуальный метод – смотрим на тип ссылки

cow.Eat(); //КОРОВА КАК КОРОВА

cowA.Eat(); //КОРОВА КАК ЖИВОТНОЕ

(cowA as Cow).Eat(); //животное к корове и , следовательно ,будет жевать сено //null

((Cow)cowA).Eat(); //аналогично //exception неправильное приведение типов

//виртуальный метод – смотрим на тип, котырым проинициализирована ссылка

Console.WriteLine("--------SLEEP--------");

cow.Sleep(); //вызов виртуального метода КОРОВА корова спит стоя

cowA.Sleep();//вызов виртуального метода ЖИВОТНОЕ корова спит стоя

Console.WriteLine("--------VOICE--------");

cow.Voice(); ///вызов метода корова мычит ( как невиртуальный)

(cow as Animal).Voice(); //приводим корову к животному и тогда корова имеет голос

((IVoice)cow).Voice(); //приведение к интерфейсу IVoice, вызов явной реализации интерфейса

//что здесь происходит

cowA.Voice();

(cowA as Cow).Voice();

((IVoice)cowA).Voice();

List<BaseAnimal> bs = new List<BaseAnimal>();

bs.Add(cow);

bs.Add(cowA);

bs.Add(new Animal());

foreach (var item in bs)

{

item.DoSomeThing(); //для всех

if (item is Animal) (item as Animal).Eat(); //только для тех у кого реализован метод Eat

if (item is Cow) (item as Cow).Eat();

}