Механизм наследования. Отношения между классами

Задание №1. Базовый класс: Животное (поля: кличка, рост – h в м) Метод: Вес животного – k \*h3 кг, где k-коэффициент. Потомок: Кошка (k= 15, поле – порода животного) Изменения в потомках: Перевести вес животного в граммы.

Листинг программы:

public class Animal

{

public int K { get; protected set; }

public string Alias { get; private set; }

public double Height { get; private set; }

public Animal(string alias, double height, int k) { Alias = alias; Height = height; K = k; }

public double GetWeight() => K \* Height;

public override string ToString() => $"Кличка: '{Alias}'; Рост: {Height}м., Вес: {GetWeight()}кг.";

}

public class Cat : Animal

{

public string Breed { get; private set; }

public Cat(string breed, string alias, double height) : base(alias, height, 15) => Breed = breed;

public double GetWeightInGrams() => GetWeight() \* 1000;

public override string ToString() => $"Кошка. Порода: '{Breed}'; {base.ToString()}; Вес в граммах: {GetWeightInGrams()}г.";

static void Main(string[] args)

{

Console.WriteLine(new Animal("Некто", 0.1, 13) + "\n" + new Cat("Шерстяная", "СобакаСутулая", 0.3));

Console.ReadLine();

}

}

Анализ программы:



Рисунок 1.1 - Анализ программы

Источник: собственная разработка

Задание №2. В рамках консольного приложения разработать класс В-

наследник класса А (класс А с полями а и b и свойством с. Свойство –

значение выражения над полями а и b (выражение и типы полей – см.

вариант в таблице 1). Поля инициализировать при объявлении класса.

Конструктор оставить по умолчанию. Проследить, чтобы поля а и b

напрямую в других классах были недоступны.) с полем d и свойством

с2. Свойство с2 – результат вычисления выражения над полями a, b,

d. В теле свойства использовать управляющий оператор (см. вариант

в таблице 1) Для класса В определить 2 конструктора: один –

наследуется от конструктора класса А, второй –собственный. В теле

программы создать объекты классов А и В, продемонстрировав

работу всех конструкторов. Вывести значения свойства на экран.

Листинг программы:

namespace ProgramSpaceTwo

{

public enum Post

{

Accountant = 164,

Engineer = 180,

Electrician = 150,

Mechanic = 154,

Agronomist = 170,

Programmer = 160,

Builder = 100

}

public class Accauntant

{

private int hours;

// конструктор

public Accauntant()

{

hours = 0;

}

public void Foreaching()

{

foreach (string post in Enum.GetNames(typeof(Post)))

{

Console.Write($"Введите, cколько времени проработал за этот месяц {post}: ");

hours = Convert.ToInt32(Console.ReadLine());

Enum.TryParse<Post>(post, true, out Post postEnum);

Console.WriteLine($"Полагается ли {post} премия: {AskForBonus(postEnum, hours)}");

Console.WriteLine();

}

}

/// <summary>

/// Решени, начислять бонус или нет

/// </summary>

/// <param name="worker"> Перечисление профессий </param>

/// <param name="hours"> Время работы работника </param>

/// <returns></returns>

public bool AskForBonus(Post worker, int hours)

{

if ((int)worker < hours)

{

return true;

}

else

{

return false;

}

}

}

public class Legacy : Accauntant

{

private int d;

// конструктор

public Legacy()

{

d = 0;

}

public int Legacy\_Result()

{

int a = 12;

int b = 0;

int oneYers = 0;

int[] enums = (int[])Enum.GetValues(typeof(Post));

int sizeEnum = Enum.GetNames(typeof(Post)).Length;

for (int i = 0; i < sizeEnum; i++)

{

d = enums[i];

oneYers += a \* (b + d);

}

return oneYers;

}

}

public class ProgramMain

{

public static void Main()

{

Accauntant acc = new Accauntant();

acc.Foreaching();

Legacy l = new Legacy();

Console.WriteLine($"{l.Legacy\_Result()} Часов в год на работников.");

}

}

}

Анализ программы:

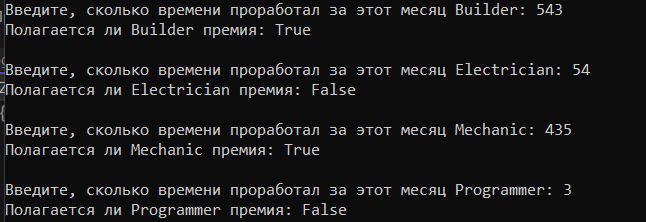


Рисунок 1.1 - Анализ программы

Источник: собственная разработка