**10 Механизм наследования. Отношения между классами**

Задание 1.

Базовый класс: Квартира (поля: название, стоимость 1м 2, площадь) Метод: Стоимость квартиры. Потомок: Квартира в центре (поле название района) Изменения в потомках: Увеличить стоимость с учетом надбавки за расположение на 0.01 стоимости квартиры.

Листинг программы:

using System;

using System.Collections.Generic;

namespace TEST

{

class Program

{

static string[] pars = new string[4];

static List<Apartment> ap1 = new List<Apartment>();

static List<ApartmentInTheCenter> ap2 = new List<ApartmentInTheCenter>();

static void Main(string[] args)

{

char temp;

while (true)

{

Console.Clear();

Console.WriteLine("Выберите действие");

Console.WriteLine(new string('-', 50));

Console.WriteLine("1) Создать объект класса Квартира");

Console.WriteLine("2) Создать объект класса Квартира в центре");

Console.WriteLine("3) Вывести все поля объектов");

Console.WriteLine("4) Вывести стоимости квартир");

Console.WriteLine("0) Выход");

Console.WriteLine(new string('-', 50));

temp = Console.ReadLine()[0];

switch (temp)

{

case '1':

First();

break;

case '2':

Second();

break;

case '3':

GetAllPars();

break;

case '4':

GetCosts();

break;

case '0':

Environment.Exit(1);

break;

default:

break;

}

}

}

static void First()

{

Console.Clear();

Console.WriteLine("Создание объекта класса Квартира");

Console.WriteLine(new string('-', 50));

Console.Write("Название: ");

pars[0] = Console.ReadLine();

Console.Write("Стоимость 1 кв. метра: ");

pars[1] = Console.ReadLine().Replace('.', ',');

Console.Write("Площадь: ");

pars[2] = Console.ReadLine().Replace('.', ',');

ap1.Add(new Apartment(pars[0], double.Parse(pars[1]), double.Parse(pars[2])));

Console.Clear();

Console.WriteLine("Успешно!");

Console.WriteLine(new string('-', 50));

Console.WriteLine("Для продолжения нажмите Enter");

Console.ReadLine();

}

static void Second()

{

Console.Clear();

Console.WriteLine("Создание объекта класса Квартира в центре");

Console.WriteLine(new string('-', 50));

Console.Write("Название района: ");

pars[0] = Console.ReadLine(); ;

Console.Write("Название квартиры: ");

pars[1] = Console.ReadLine();

Console.Write("Стоимость 1 кв. метра: ");

pars[2] = Console.ReadLine().Replace('.', ',');

Console.Write("Площадь: ");

pars[3] = Console.ReadLine().Replace('.', ',');

ap2.Add(new ApartmentInTheCenter(pars[0], pars[1], double.Parse(pars[2]), double.Parse(pars[3])));

Console.Clear();

Console.WriteLine("Успешно!");

Console.WriteLine(new string('-', 50));

Console.WriteLine("Для продолжения нажмите Enter");

Console.ReadLine();

}

static void GetAllPars()

{

Console.Clear();

Console.WriteLine("Поля объектов");

Console.WriteLine(new string('-', 50));

Console.WriteLine("Квартиры:");

foreach (var x in ap1)

{

Console.WriteLine("-{0}:", x.Name);

Console.WriteLine("--Стоимость 1 кв. метра: {0}", x.CostPerSquareMeter);

Console.WriteLine("--Площадь: {0}", x.Square);

}

Console.WriteLine("Квартиры в центре:");

foreach (var x in ap2)

{

Console.WriteLine("-{0}:", x.Name);

Console.WriteLine("--Район: {0}", x.DistrictName);

Console.WriteLine("--Стоимость 1 кв. метра: {0}", x.CostPerSquareMeter);

Console.WriteLine("--Площадь: {0}", x.Square);

}

Console.WriteLine(new string('-', 50));

Console.WriteLine("Для продолжения нажмите Enter");

Console.ReadLine();

}

static void GetCosts()

{

Console.Clear();

Console.WriteLine("Стоимости квартир");

Console.WriteLine(new string('-', 50));

Console.WriteLine("Квартиры:");

foreach (var x in ap1)

{

Console.WriteLine("-{0}: {1}", x.Name, x.GetApartmentCost());

}

Console.WriteLine("Квартиры в центре:");

foreach (var x in ap2)

{

Console.WriteLine("-{0}: {1}", x.Name, x.GetApartmentCost());

}

Console.WriteLine(new string('-', 50));

Console.WriteLine("Для продолжения нажмите Enter");

Console.ReadLine();

}

}

class Apartment

{

private string name;

private double costPerSquareMeter;

private double square;

public string Name { get => name; set => name = value; }

public double CostPerSquareMeter { get => costPerSquareMeter; set => costPerSquareMeter = value; }

public double Square { get => square; set => square = value; }

public Apartment(string name, double cost, double square)

{

Name = name;

CostPerSquareMeter = cost;

Square = square;

}

public virtual string[] WritePars() => new[] { Name, CostPerSquareMeter.ToString(), Square.ToString() };

public virtual double GetApartmentCost() => CostPerSquareMeter \* Square;

}

class ApartmentInTheCenter : Apartment

{

private string districtName;

public string DistrictName { get => districtName; set => districtName = value; }

public ApartmentInTheCenter(string distName, string name, double cost, double square) : base(name, cost, square){

DistrictName = distName;}

public override string[] WritePars() => new[] { DistrictName, Name, CostPerSquareMeter.ToString(), Square.ToString() };

public override double GetApartmentCost() => base.GetApartmentCost() \* 1.01;}}

Таблица 10.1 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| Квартира  200  159  Люблино  Пушкина  500  250 | Поля объектов  --------------------------------------------------  Квартиры:  -Квартира :  --Стоимость 1 кв. метра: 200  --Площадь: 159  Квартиры в центре:  -Пушкина:  --Район: Люблино  --Стоимость 1 кв. метра: 500  --Площадь: 250  --------------------------------------------------  Для продолжения нажмите Enter |

Анализ результатов представлен на рисунке 10.1:

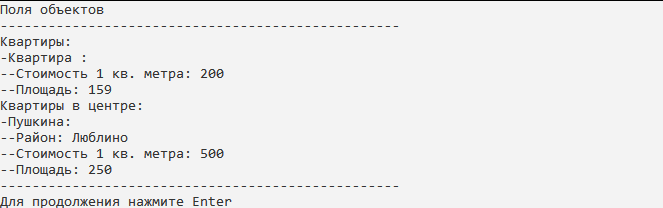


Рисунок 1.1 – Результаты работы программы

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 bнапрямую в других классах были недоступны.) с полем d и свойство мс2. 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 телепрограммы создать объекты классов А и В, продемонстрировав работу всех конструкторов. Вывести значения свойства на экран. Switch.

Листинг программы:

using System;

using System.Collections.Generic;

using System.Text;

namespace ConsoleApplication10

{

class A

{

public int a;

public int b;

public A() { }

public A(int a, int b)

{

this.a = a;

this.b = b;

}

}

class B : A

{

private int d;

public int c2

{

get

{

switch (a)

{

case 0:

return 0;

break;

}

switch (b)

{

case 0:

return 0;

break;

default:

return a + b;

break;

}

}

}

public B() { }

public B(int a, int b, int d) : base(a, b)

{

this.d = d;

}

}

class Program

{

static void Main(string[] args)

{

A a = new A();

A aa = new A(1, 2);

B b = new B();

B bb = new B(1, 2, 3);

Console.WriteLine(bb.c2);

}

}

}

Таблица 10.2 – Входные и выходные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  | 3 |

Анализ результатов представлен на рисунке 10.2:



Рисунок 10.2 – Результаты работы программы

Источник: собственная разработка