**10. Механизм наследование. Отношение между классами**

Задание 1. Наследование

Составить программу с одним родительским классом и потомком. Все поля должны быть закрытыми. Базовый класс должен содержать конструкторы с параметрами, методы доступа к закрытым полям, вывод полей и указанный метод. Производный класс содержит дополнения и изменения, организовать вывод новых полей потомка, при этом имена методов совпадают с именами методов базового класса. Составить тестирующую программу с выдачей результатов. Создать объекты базового и производного типов. В программе должна выполняться проверка всех разработанных элементов класса.

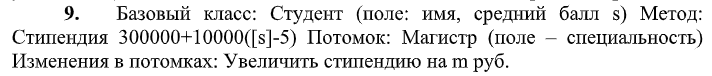


Рисунок 10.1 – Условие задания

Источник: собственная разработка

Листинг программы:

using System;

namespace zadanie1.\_1

{

class Demo

{

static void Main(string[] args)

{

//Задаем входящие параметры

Master ob = new Master("Программист", "Петров", 4.5);

//выполнение

Console.WriteLine(ob.Specialty + " " + ob.Name + " " + ob.Avg\_rating);

Console.WriteLine("Ваша стипендия");

double stn = Convert.ToDouble(Console.ReadLine());

Console.WriteLine("Насколько повысилась стипендия?");

double stu = Convert.ToDouble(Console.ReadLine());

Console.WriteLine("Увеличенная стипендия составила \n" + ob.UpGrant(ob.Grant(stn), stu));

Console.ReadLine();

}

}

}

class Student

{

string name;

double avg\_rating;

//создаем свойства для доступа к закрытым полям класса

public string Name

{

get { return name; }

set { name = value; }

}

public double Avg\_rating

{

get { return avg\_rating; }

set { avg\_rating = value; }

}

//Создаем конструктор класса используя в нем открытые свойства

public Student(string n, double r)

{

Name = n;

Avg\_rating = r;

}

public double Grant(double g)

{

return g;

}

}

class Master : Student

{

string specialty;

//Такое же св-во для доступа к закрытым полям

public string Specialty

{

get { return specialty; }

set { specialty = value; }

}

//Cоздаем конструктор класса магистр который принимает свой дополнительный параметр специальность и наследует два параметра из конструктора класса Студент

public Master(string sp, string n, double r)

: base(n, r)

{

Specialty = sp;

}

public double UpGrant(double g, double m)

{

double up = g + m;

return up;

}

}

Таблица 10.1 – Выходные и входные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 1000, 500 | 1500 |

Анализ результатов:

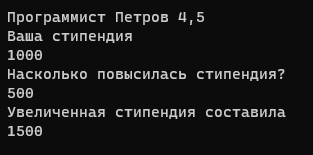


Рисунок 10.2 – Результат работы программы

Источник: собственная разработка

Задание 2. Перечисления

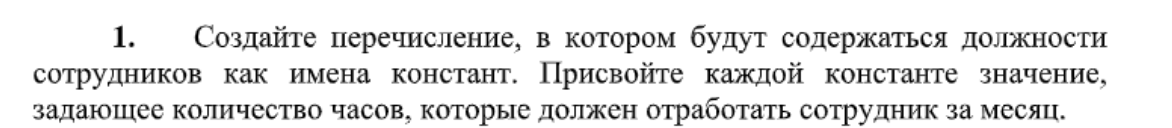


Рисунок 10.3 – Условие задания

Источник: собственная разработка

Листинг программы:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace zadanie2

{

class Program

{

public static void Main()

{

B b = new B();

Console.WriteLine("a={0} ", b.a, "b={0} ", b.b, "d={0} ", b.d);

Console.WriteLine("2={0} ", b.c, "c2={0} ", b.c2);

}

}

internal class B

{

internal object b;

internal int a;

internal int c;

internal object c2;

internal object d;

}

}

Таблица 10.2 – Выходные и входные данные

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| A, 2 | a=0, 2=0 |

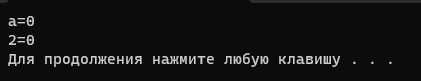
Анализ результатов:

Рисунок 10.4 – Результат работы программы

Источник: собственная разработка