**2.Классы**

Задание №1. Создать класс А с целочисленными полями а и b и двумя методами согласно варианту. Внутри класса реализовать конструктор для инициализации a и b. Создать объект класса и продемонстрировать работу со всеми элементами класса.

Метод нахождения среднего арифметического a и b, метод вычисления значения выражения b3+

Листинг программы:

Файл Program:

using z1.Models;

Console.Write("Введите а: ");

int numA = Convert.ToInt32(Console.ReadLine());

Console.Write("Введите b: ");

int numB = Convert.ToInt32(Console.ReadLine());

A a = new A(numA, numB);

Console.WriteLine($"вызов метода среднего арифметического чисел: {a.GetArithmeticMean()}");

Console.WriteLine($"вызов метода пример чисел: {a.GetValueExample():.###}");

Файл A:

namespace z1.Models

{

public class A

{

private int \_a;

private int \_b;

public A(int a, int b)

{

\_a = a;

\_b = b;

}

public double GetArithmeticMean()=> (\_a + \_b) / 2;

public double GetValueExample()=> Math.Pow(\_b,3) + Math.Sqrt(\_a);

}

}

Таблица 2.1 – Входные и выходные данные программы задание №1

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| numA:4  numB:6 | 5; 218 |

Анализ результатов:

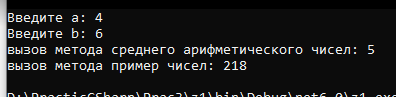


Рисунок 2.1 – Результат выполнения программы задание №1

Задание №2. Построить иерархию классов в соответствии с вариантом задания: Организация, страховая компания, нефтегазовая компания, завод.

Листинг программы:

Файл Program:

using z2.Models;

Factory factory = new Factory("Савушкин","Сидоров", 123122);

factory.Work();

OilCompany oilCompany = new OilCompany("БелНефть", "Иванов", 12412214);

oilCompany.PumpOil();

InsuranceСompany insuranceСompany = new InsuranceСompany("БелСтрах", "Тутчев", 124124124);

insuranceСompany.InsureOrgan(oilCompany);

Файл Organization:

namespace z2.Models

{

public abstract class Organization

{

private string \_name;

private string \_fullNameOwner;

private decimal \_profit;

public string Name { get { return \_name; } set {\_name = value; } }

public string FullNameOwner { get {return \_fullNameOwner; } set {\_fullNameOwner = value; } }

public decimal Profit { get {return \_profit; } set {\_profit = value; } }

protected Organization(string name, string fullNameOwner, decimal profit)

{

\_name = name;

\_fullNameOwner = fullNameOwner;

\_profit = profit;

}

}

}

Файл Factory:

namespace z2.Models

{

public class Factory : Organization

{

public Factory(string name, string fullNameOwner, decimal profit) : base(name, fullNameOwner, profit)

{

}

public void Work()

{

Console.WriteLine("Завод работает");

}

}

}

Файл InsuranceСompany:

namespace z2.Models

{

public class InsuranceСompany : Organization

{

public InsuranceСompany(string name, string fullNameOwner, decimal profit) : base(name, fullNameOwner, profit)

{

}

public void InsureOrgan(Organization organization)

{

Console.WriteLine($"Старх. компания страхует {organization.Name}");

}

}

}

Файл OilCompany:

namespace z2.Models

{

public class OilCompany : Organization

{

public OilCompany(string name, string fullNameOwner, decimal profit) : base(name, fullNameOwner, profit)

{

}

public void PumpOil()

{

Console.WriteLine("Нефт. комп. качает нефть");

}

}

}

Таблица 2.2 – Входные и выходные данные программы задание №2

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  | Завод работает  Нефт. комп. качает нефть  Старх. компания страхует БелНефть |

Анализ результатов:



Рисунок 2.2 – Результат выполнения программы задание №2

Задание №3. Составить описание класса для вектора, заданного координатами его концов в трехмерном пространстве. Обеспечить операции сложения и вычитания векторов с получением нового вектора (суммы или разности), вычисления скалярного произведения двух векторов, длины вектора, косинуса угла между векторами.

Написать программу, демонстрирующую все разработанные элементы класса.

Листинг программы:

Файл Program:

using z3.Models;

Vector3 vector1 = new Vector3(1, 2, 3);

Console.WriteLine($"Длина вектора: {vector1.Length:.###}");

Vector3 vector2 = new Vector3(4, 5, 6);

Console.WriteLine($"Скалярное проеизведение двух векторов: {vector1.GetScalarProductOfVector3(vector2)}");

Console.WriteLine($"Косинус угла между двумя веторами: {vector1.GetCosCornerBetweenVector3(vector2):.###}");

Файл Vector3:

using static System.Math;

namespace z3.Models

{

public class Vector3

{

private double \_x, \_y, \_z;

public double X { get => \_x; set => \_x = value; }

public double Y { get => \_y; set => \_y = value; }

public double Z { get => \_z; set => \_z = value; }

public double Length

{

get => Sqrt(Pow(\_x, 2) + Pow(\_y, 2) + Pow(\_z, 2));

}

public Vector3(double x, double y, double z)

{

\_x = x;

\_y = y;

\_z = z;

}

public static Vector3 operator +(Vector3 v1, Vector3 v2) => new Vector3(v1.\_x + v2.\_x, v1.\_y + v2.\_y, v1.\_z + v2.\_z);

public static Vector3 operator -(Vector3 v1, Vector3 v2) => new Vector3(v1.\_x - v2.\_x, v1.\_y - v2.\_y, v1.\_z - v2.\_z);

public double GetScalarProductOfVector3(Vector3 v) => \_x \* v.X + \_y \* v.Y + \_z \* v.Z;

public double GetCosCornerBetweenVector3(Vector3 v) => GetScalarProductOfVector3(v) / (Length \* v.Length);

}

}

Таблица 2.3 – Входные и выходные данные программы задание №3

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
|  | Длина вектора: 3,742  Скалярное проеизведение двух векторов: 32  Косинус угла между двумя веторами: ,975 |

Анализ результатов:

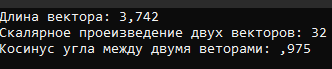


Рисунок 2.3 – Результат выполнения программы задание №3