Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет

информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра Информатики

Дисциплина «Программирование»

**ОТЧЕТ**

к лабораторной работе №4

на тему:

**«Конструкторы. Статические члены класса.**

**Шаблон проектирования Singleton.»**

БГУИР 6-05-0612-02 75

|  |
| --- |
| Выполнил студент группы 453503  МИХАЛКО Александр Николаевич |
|  |
| (дата, подпись студента) |
| Проверил ассистент каф. Информатики  РОМАНЮК Максим Валерьевич |
|  |
| (дата, подпись преподавателя) |

Минск 2025

# 1 Индивидуальное задание

**Задание 1. Вариант 6** Предметная область: Отдел кадров-Норма выработки. В классе хранить информацию о наименовании предприятия, числе работников, норму выработки часов в месяц (одинаковая для всех работников - класс Норма выработки), оплату за час, подоходный налог. Реализовать метод для подсчета общей выплаты по подоходному налогу. Реализовать возможность изменения (увеличения и уменьшения) нормы выработки.

# 2 Выполнение работы

Для выполнения задания в проект были добавлены два класса Enterprise и ProductionNorm (cм. рисунок 1).

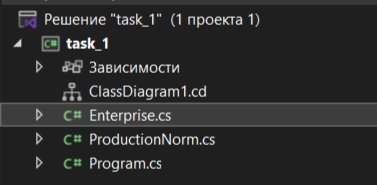


Рисунок 1 – Обозреватель решений

Был реализован класс ProductionNorm, в котором имеется свойство MonthlyHoursNorm, которое отвечает за норму выработки часов в месяц. Для MonthlyHoursNorm были реализован метод IncreaseNorm, который позволяет изменять значение поля MonthlyHoursNorm.

namespace task\_1\_

{

public class ProductionNorm

{

public int MonthlyHoursNorm { get; private set; }

public ProductionNorm()

{

MonthlyHoursNorm = 160;

}

public void IncreaseNorm(int hours)

{

MonthlyHoursNorm += hours;

}

}

}

Рассмотрим реализацию класса Enterprise, который хранит поля характеризующие предприятие: Name, EmployeeCount, PaymentPerHour, IncomeTax, Norm. Также используются методы CalculateTotalIncomeTax() для вычиления общей выплаты по подоходному налогу, метод InfoMonthlyHoursNorm для возвращения нормы выработки и GetInfo() для вывода всех полей класса. Был применен шаблон проектирования Singleton, который гарантирует, что для класс будет создан только один объект

namespace task\_1\_

{

class Enterprise

{

private static Enterprise instance;

private ProductionNorm norm;

public string Name { get; set; }

public int EmployeeCount { get; set; }

public double PaymentPerHour { get; set; }

public double IncomeTaxRate { get; set; }

public Enterprise(string name, int employeeCount, double paymentPerHour, double incomeTaxRate)

{

Name = name;

EmployeeCount = employeeCount;

PaymentPerHour = paymentPerHour;

IncomeTaxRate = incomeTaxRate;

norm = new ProductionNorm();

}

public static Enterprise GetInstance(string name, int employeeCount, double paymentPerHour, double incomeTaxRate)

{

if (instance == null)

{

instance = new Enterprise(name, employeeCount, paymentPerHour, incomeTaxRate);

}

return instance;

}

public double CalculateTotalIncomeTax()

{

return norm.MonthlyHoursNorm \* PaymentPerHour \* EmployeeCount \* IncomeTaxRate;

}

public double InfoMonthlyHoursNorm()

{

return norm.MonthlyHoursNorm;

}

public void ChangeNorm(int hours)

{

norm.IncreaseNorm(hours);

}

public void GetInfo()

{

Console.WriteLine($"Предприятие: {Name}");

Console.WriteLine($"Количество работников: {EmployeeCount}");

Console.WriteLine($"Оплата за час: {PaymentPerHour}");

Console.WriteLine($"Подоходный налог: {IncomeTaxRate}");

Console.WriteLine($"Норму выработки часов в месяц: {norm.MonthlyHoursNorm}");

Console.WriteLine($"Общая выплата по подоходному налогу: {CalculateTotalIncomeTax()}");

}

}

}

Покажем принцип работы этой программы в классе Program, где, помимо всего, реализованы методы InputString(), InputInt(), InputDoublePaymen(), InputDoubleTax(), InputDecisionYesNo(), InputDecisionChanges(), CorrectFinishAnswer() для проверки на корректный ввод.

namespace task\_1\_

{

internal class Program

{

static void Main(string[] args)

{

Run();

}

static void Run()

{

Console.WriteLine("6 Вариант");

Console.WriteLine();

Enterprise enterprise;

while (true)

{

Console.WriteLine("Введите название предприятия: ");

string name = InputString();

Console.WriteLine("Введите количество работников: ");

int employeeCount = InputInt();

Console.WriteLine("Введите оплату за час работника: ");

double paymentPerHour = InputDoublePaymen();

Console.WriteLine("Введите подоходный налог с каждого работника в %: ");

double incomeTax = InputDoubleTax() / 100;

enterprise = new Enterprise(name, employeeCount, paymentPerHour, incomeTax);

Console.WriteLine($"Сейчас норма выработки часов в месяц: {enterprise.InfoMonthlyHoursNorm()}");

Console.WriteLine("Изменить норму? (введиет число: 1 - да, 0 - нет): ");

bool decision = InputDecisionYesNo();

if (decision)

{

Console.WriteLine("Введите на сколько вы хотите изменить норму: ");

int changes = InputDecisionChanges();

enterprise.ChangeNorm(changes);

}

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

enterprise.GetInfo();

Console.WriteLine();

Console.WriteLine("Do you want to continue or stop[Y/n]");

string decicion = Console.ReadLine() ?? string.Empty;

bool result = CorrectFinishAnswer(decicion);

if (!result) break;

}

}

static string InputString()

{

while (true)

{

string value = Console.ReadLine() ?? string.Empty;

value = value.Trim();

if (string.IsNullOrEmpty(value))

{

Console.WriteLine("Пустая строка без символов, введите значение");

}

else

{

return value;

}

}

}

static int InputInt()

{

string value;

int getValueInInt;

bool valid = false;

while (!valid)

{

value = Console.ReadLine() ?? string.Empty;

valid = int.TryParse(value, out getValueInInt);

if (valid == false || getValueInInt <= 0)

{

Console.WriteLine("число работников это натуральное число");

valid = false;

}

else

{

return getValueInInt;

}

}

return 0;

}

static double InputDoublePaymen()

{

string value;

double getValueInDouble;

bool valid = false;

while (!valid)

{

value = Console.ReadLine() ?? string.Empty;

valid = double.TryParse(value, out getValueInDouble);

if (valid == false || getValueInDouble < 0)

{

Console.WriteLine("оплата должна быть неотрицательным числом");

valid = false;

}

else

{

return getValueInDouble;

}

}

return 0.0;

}

static double InputDoubleTax()

{

string value;

double getValueInDouble;

bool valid = false;

while (!valid)

{

value = Console.ReadLine() ?? string.Empty;

valid = double.TryParse(value, out getValueInDouble);

if (valid == false || getValueInDouble < 0 || getValueInDouble > 100)

{

Console.WriteLine("процент должен быть в пределах от 0 до 100");

valid = false;

}

else

{

return getValueInDouble;

}

}

return 0.0;

}

static bool InputDecisionYesNo()

{

string value;

int getValueInInt;

bool valid = false;

while (!valid)

{

value = Console.ReadLine() ?? string.Empty;

valid = int.TryParse(value, out getValueInInt);

if (valid == false || (getValueInInt != 0 && getValueInInt != 1))

{

Console.WriteLine("вы должны ввести число 0 или 1");

valid = false;

}

else

{

if (getValueInInt == 0) return false;

else return true;

}

}

return false;

}

static int InputDecisionChanges()

{

string value;

int getValueInInt;

bool valid = false;

while (!valid)

{

value = Console.ReadLine() ?? string.Empty;

valid = int.TryParse(value, out getValueInInt);

if (valid == false)

{

Console.WriteLine("введите число");

valid = false;

}

else

{

return getValueInInt;

}

}

return 0;

}

static bool CorrectFinishAnswer(string answer)

{

while (true)

{

// Удаляем лишние пробелы в начале и конце строки

answer = answer.Trim();

if (string.IsNullOrEmpty(answer))

{

return true;

}

if (answer == "Y" || answer == "y") return true;

else if (answer == "N" || answer == "n") return false;

else

{

Console.WriteLine("It is not valid answer, try again:");

answer = Console.ReadLine() ?? string.Empty;

}

}

}

}

}

Результат работы программы продемонстрирован ниже (см. рисунок 2).

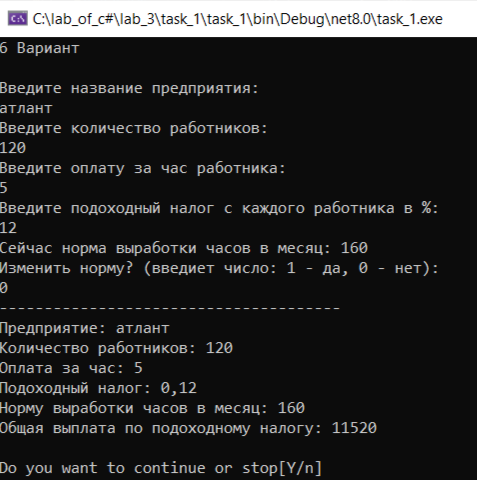


Рисунок 2 – Результат работы программы

# Вывод

В ходе лабораторной работы были изучены назначения конструкторов в классе, перегрузка методов класса, возможности членов класса с модификатором static, шаблоны проектирования Singleton, посредством написания программы, описывающей предприятие.