|  |  |
| --- | --- |
| КГЭУ | МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования** **«КАЗАНСКИЙ ГОСУДАРСТВЕННЫЙ ЭНЕРГЕТИЧЕСКИЙ УНИВЕРСИТЕТ»** (ФГБОУ ВО «КГЭУ») |

**Институт цифровых технологий и экономики**

**Кафедра «Цифровые системы и модели»**

# **Дисциплина: Объектно-ориентированное программирование**

**Отчет о лабораторной работе № 2**

**Классы. Поля. Методы. Конструкторы.**

**Исполнитель:** Тазеев Р.Р.

**Группа:** ТРП-1-23

**Проверяющий:** Халидов А.А.

**Должность:** доцент кафедры ЦСМ

**КАЗАНЬ – 2024**

**Введение**

В данной лабораторной работе рассматриваются ключевые аспекты объектно-ориентированного программирования (ООП): классы, поля, методы и конструкторы. Эти элементы составляют основу ООП и обеспечивают модульность, повторное использование кода и более эффективное управление сложностью программ.

Цель работы — изучить процесс создания и взаимодействия объектов, исследовать механизм работы конструкторов, а также закрепить навыки использования методов для обработки данных, хранимых в полях объектов. Выполнение заданий лабораторной работы позволит студентам глубже понять, как разрабатываются и функционируют программные системы на основе классов.

Понимание принципов работы классов, полей, методов и конструкторов является важным шагом на пути освоения более сложных концепций ООП, таких как наследование, полиморфизм и инкапсуляция.

**Вариант 27**

**Задача 1.** Создать класс Numbers. Закрытые поля класса целого типа – а, b. Реализовать конструкторы с параметрами и без параметров, свойства и методы:  
- Метод NewK для задания новых значений полей.

- Метод Print для вывода значений полей на экран.

- Метод без параметров Nod, который вычисляет наибольший общий делитель этих чисел.

- Метод без параметров Nok, который вычисляет наименьшее общее кратное этих чисел.

Код:

class Numbers

    {

        private int a, b;

        public Numbers() { a = 0; b = 0; }

        public Numbers(int a, int b)

        {

            this.a = a;

            this.b = b;

        }

        public int A

        {

            get { return a; }

            set { a = value; }

        }

        public int B

        {

            get { return b; }

            set { b = value; }

        }

        public void NewK(int a, int b)

        {

            this.a = a;

            this.b = b;

        }

        public void Print()

        {

            Console.WriteLine($"A = {a}, B = {b}");

        }

        public int Nod()

        {

            int x = a, y = b;

            while (x != y)

            {

                if (x > y) x -= y;

                else y -= x;

            }

            return x;

        }

        public int Nok()

        {

            return a \* b / Nod();

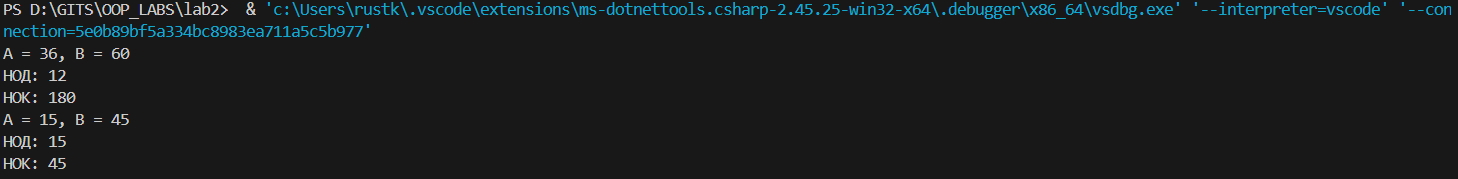
        }

    }

Код проверки:  
Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Вывод:



**Задача 2.** Разработать класс для указанного объекта. Включить в класс свойства со средствами доступа **set**, **get**, в которых обеспечить корректную работу с полями. Определить метод вывода значений полей. **Student**: Фамилия, Имя, Отчество, Дата рождения, Адрес, Телефон, Факультет, Курс. В методе **Main()** создать массив объектов. Вывести:

a) список студентов заданного факультета;

б) списки студентов для каждого факультета и курса;

в) список студентов, родившихся после заданного года.

Код:

class Student

    {

        public string LastName { get; set; }

        public string FirstName { get; set; }

        public string MiddleName { get; set; }

        public DateTime BirthDate { get; set; }

        public string Address { get; set; }

        public string Phone { get; set; }

        public string Faculty { get; set; }

        public int Course { get; set; }

        public void Print()

        {

            Console.WriteLine($"{LastName} {FirstName} {MiddleName}, {BirthDate.ToShortDateString()}, {Faculty}, {Course}");

        }

        public static void FilterByFaculty(Student[] students, string faculty)

        {

            var filtered = students.Where(s => s.Faculty == faculty);

            foreach (var student in filtered) student.Print();

        }

        public static void FilterByYear(Student[] students, int year)

        {

            var filtered = students.Where(s => s.BirthDate.Year > year);

            foreach (var student in filtered) student.Print();

        }

    }

Код проверки:

 // Часть 2: Проверка класса Student

        Student[] students = new Student[]

        {

            new Student { LastName = "Иванов", FirstName = "Иван", MiddleName = "Иванович", BirthDate = new DateTime(2000, 5, 15), Faculty = "Физика", Course = 3 },

            new Student { LastName = "Петров", FirstName = "Петр", MiddleName = "Петрович", BirthDate = new DateTime(1999, 8, 22), Faculty = "Математика", Course = 2 },

            new Student { LastName = "Сидоров", FirstName = "Сидор", MiddleName = "Сидорович", BirthDate = new DateTime(2001, 12, 5), Faculty = "Физика", Course = 1 },

            new Student { LastName = "Кузнецов", FirstName = "Алексей", MiddleName = "Алексеевич", BirthDate = new DateTime(2002, 3, 18), Faculty = "Математика", Course = 3 }

        };

        Console.WriteLine("\nСтуденты факультета 'Физика':");

        Student.FilterByFaculty(students, "Физика");

        Console.WriteLine("\nСтуденты, родившиеся после 2000 года:");

        Student.FilterByYear(students, 2000);

Вывод:

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

**Задача №3.** Создать класс Rectangle, содержащий следующие члены класса:

- поля: int a, b;

- конструктор, позволяющий создать экземпляр класса с заданными длинами сторон;

- методы, позволяющие вывести длины сторон прямоугольника на экран, рассчитать

периметр прямоугольника, рассчитать площадь прямоугольника;

- свойства, позволяющие получить/установить длины сторон прямоугольника (доступное

для чтения и записи), установить, является ли данный прямоугольник квадратом (доступное

только для чтения);

Написать программу, демонстрирующую все разработанные элементы класса.

Код:

class Rectangle

    {

        private int a, b;

        public Rectangle(int a, int b)

        {

            this.a = a;

            this.b = b;

        }

        public int A

        {

            get { return a; }

            set { a = value; }

        }

        public int B

        {

            get { return b; }

            set { b = value; }

        }

        public int Perimeter()

        {

            return 2 \* (a + b);

        }

        public int Area()

        {

            return a \* b;

        }

        public bool IsSquare

        {

            get { return a == b; }

        }

        public void Print()

        {

            Console.WriteLine($"A = {a}, B = {b}, Perimeter = {Perimeter()}, Area = {Area()}, IsSquare = {IsSquare}");

        }

    }

Код проверки:

 Rectangle rect1 = new Rectangle(5, 10);

        rect1.Print();

        rect1.A = 8;

        rect1.B = 8;

        rect1.Print();

Вывод:



**Задача №4.** Разрабатываемый класс должен, как правило, содержать следующие элементы: скрытые

поля, конструкторы с параметрами и без параметров, методы, свойства. Методы и свойства

должны обеспечивать непротиворечивый, полный, минимальный и удобный интерфейс

класса. При возникновении ошибок предусмотреть вывод сообщения.

Описать класс, представляющий треугольник. Предусмотреть методы для создания

объектов, вычисления площади, периметра и точки пересечения медиан. Описать свойства

для получения состояния объекта. При невозможности построения треугольника

предусмотреть вывод сообщения.

Написать программу, демонстрирующую все разработанные элементы класса.

Код:

 class Triangle

    {

        private int a, b, c;

        public Triangle(int a, int b, int c)

        {

            if (a + b <= c || a + c <= b || b + c <= a)

                throw new ArgumentException("Невозможно построить треугольник с заданными сторонами.");

            this.a = a;

            this.b = b;

            this.c = c;

        }

        public int Perimeter()

        {

            return a + b + c;

        }

        public double Area()

        {

            double s = Perimeter() / 2.0;

            return Math.Sqrt(s \* (s - a) \* (s - b) \* (s - c));

        }

        public (double, double) MedianIntersection()

        {

            return ((a + b + c) / 3.0, (a + b + c) / 3.0);

        }

        public void Print()

        {

            Console.WriteLine($"A = {a}, B = {b}, C = {c}, Perimeter = {Perimeter()}, Area = {Area()}");

        }

    }

Код для проверки:

try

        {

            Triangle triangle1 = new Triangle(3, 4, 5);

            triangle1.Print();

            Console.WriteLine($"Точка пересечения медиан: {triangle1.MedianIntersection()}");

            // Треугольник с неправильными сторонами

            Triangle triangle2 = new Triangle(1, 2, 8);  // Ошибка

        }

        catch (ArgumentException ex)

        {

            Console.WriteLine(ex.Message);

        }

Вывод:

Изображение выглядит как текст, Шрифт, снимок экрана, линия

Автоматически созданное описание

**Ссылка на файлы лабораторных работ**: <https://github.com/ironsast/OOP_LABS>

**Вывод:**

В ходе выполнения лабораторной работы были детально изучены такие основные элементы объектно-ориентированного программирования, как классы, поля, методы и конструкторы. Полученные знания и практический опыт позволили лучше понять принципы создания объектов, а также взаимодействие между элементами класса.

Закрепление этих навыков является важным шагом для освоения более сложных аспектов ООП, что открывает возможности для дальнейшего эффективного использования объектно-ориентированных технологий в разработке программного обеспечения.