

Лабораторная работа №5

Тема: Разработка классов, создание объектов и использование их в программах.

Цель: Научиться разрабатывать классы, создавать объекты и использовать их в программах.

Выполнение работы:

Пример 1. Создать класс, моделирующий работу с треугольником (задаются три стороны **a**, **b** и **c** и вычисляется площадь треугольника **s**).

Возможная реализация программы:

```
using System;
namespace Prim_Class0
{
    class Treug
    {
        double a, b, c; // стороны треугольника
        // метод для ввода данных
        public void vvod()
        {
            Console.Write("a=");
            a = double.Parse(Console.ReadLine());
            Console.Write("b=");
            b = double.Parse(Console.ReadLine());
            Console.Write("c=");
            c = double.Parse(Console.ReadLine());
        }
        // вычисление площади по формуле Герона
        public double Pl()
        {
            double s = 0;
            if (YesTreug())
            {
                double p = (a + b + c) / 2;
                s = Math.Sqrt(p * (p - a) * (p - b) * (p - c));
            }
            else
```

```

{
Console.WriteLine("Треугольник не
существует.");
s = -1;
}
return s;
}
// Закрытый метод для проверки существования треугольника
private bool YesTreug()
{
if(a + b > c && a + c > b && b + c > a)
return true;
else
return false;
}
}
class Program
{
public static void Main(string[] args)
{
Console.WriteLine("Тестирование класса Treug (треугольник)");
Treug x = new Treug();
x.vvod();
double s = x.Pl();
Console.WriteLine("Площадь треугольника s={0}", s);
Console.Write("Press any key to continue . .
. ");
Console.ReadKey(true);
}
}
}

```

```

Тестирование класса Treug (треугольник)
a=3
b=4
c=5
Площадь треугольника s=6

```

Рисунок 1 - Результаты работы программы

Пример 2. Для класса, моделирующего работу с окружностью, создадим свойство для изменения радиуса окружности.

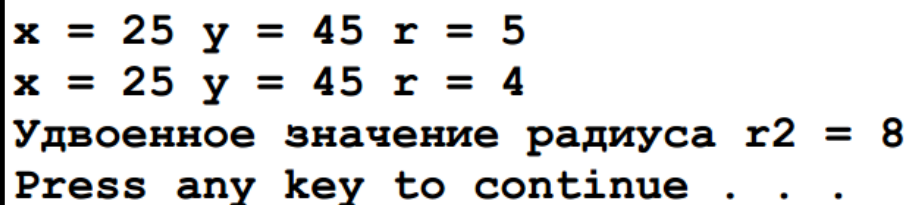
Возможный текст программы:

```
using System;
namespace Prim_Svojstvo
{
    public class Okr
    {
        int x, y, r; // координаты центра окружности и
        её радиус
        public Okr()
        {
            x = y = 100;
            r = 10;
        }
        public Okr(int x0, int y0, int r0)
        {
            x = x0;
            y = y0;
            r = r0;
        }
        public void print()
        {
            Console.WriteLine("x = {0} y = {1} r = {2}",
            x, y, r);
        }
        public int Radius
        {
            set
            {
                if(value > 0)
                    r = value;
                else
                {
                    Console.WriteLine("Недопустимое значение для поля: {0}", value);
                    Console.WriteLine("Поле сохраняет прежнее значение: {0}", r);
                }
            }
        }
    }
}
```

```

    }
    get
    {
        return r;
    }
}
class Program
{
    public static void Main(string[] args)
    {
        Okr x = new Okr(25, 45, 5);
        x.print();
        x.Radius = 4;
        x.print();
        int r2 = x.Radius * 2;
        Console.WriteLine("Удвоенное значение радиуса r2 = {0}", r2);
        Console.Write("Press any key to continue . . . ");
        Console.ReadKey(true);
    }
}

```



```

x = 25 y = 45 r = 5
x = 25 y = 45 r = 4
Удвоенное значение радиуса r2 = 8
Press any key to continue . . .

```

Рисунок 2 - Результаты работы программы

Пример 3. Описание класса и создание объектов

```

using System;
// Описание класса:
class MyClass{
// Целочисленное поле:
public int number;
// Символьное поле:

```

```

public char symbol;
// Метод:
public void show(){
// Отображение значения целочисленного поля:
Console.WriteLine("Целочисленное поле: "+number);
// Отображение значения символьного поля:
Console.WriteLine("Символьное поле: "+symbol);
}
}
// Класс с главным методом:
class UsingObjsDemol
// Главный метод:
static void Main(){
// Первый объект:
MyClass A=new MyClass() ;
// Объектная переменная:
MyClass B;
// Второй объект:
B=new MyClass();
// Присваивание значений полям первого объекта: A.number=123;
symbol='A';
// Присваивание значений полям второго объекта:
number=321;
B.symbol='B';
// Вызов методов:
Console.WriteLine("Первый объект");
A.show();
Console.WriteLine("Второй объект");
B.show();
}
}

```

Пример 4. Присваивание объектов.

```

using System;
// Описание класса:
class MyClass{
// Целочисленное поле: public int number;

```

```

// Метод для отображения значения поля:
public void show(){
Console.WriteLine("Значение поля: "+number);
}
}
// Класс с главным методом:
class AnotherObjsDemo{
// Главный метод:
static void Main(){
// Объектные переменные:
MyClass A, B;
// Создание объекта:
A=new MyClass();
// Присваивание объектных переменных:
B=A;
// Присваивание значения полю через первую
// объектную переменную:
A.number=123;
// Вызов метода через вторую объектную переменную:
B.show();
// Присваивание значения полю через вторую
// объектную переменную:
B.number=321;
// Вызов метода через первую объектную переменную:
A.show();
}
}

```

Пример 5. Закрытые члены класса и перегрузка методов.

```

using System;
// Описание класса:
class MyClassf
// Закрытое целочисленное поле:
private int number;
// Закрытое символьное поле:
private char symbol;
// Открытый метод для отображения значения полей:

```

```
public void show(){
    Console.WriteLine("Поля объекта: "+number+" и "+symbol);
}
// Открытый метод для присваивания значений полям.
// Версия с двумя аргументами:
public void set(int n, char s){
    number=n;
    // Значение целочисленного поля
    symbol=s;
    // Значение символьного поля
}
// Открытый метод для присваивания значений полям.
// Версия с одним целочисленным аргументом:
public void set(int n){
    number=n; // Значение целочисленного поля
    symbol='B'; // Значение символьного поля
}
// Открытый метод для присваивания значений полям.
// Версия без аргументов:
public void set(){
    // Вызов версии метода с двумя аргументами:
    set(100,'A');
}
}
// Главный класс:
class MethodsDemo{
    // Главный метод:
    static void Main(){
        // Создание объекта:
        MyClass obj=new MyClass();
        // Присваивание значений полям:
        obj.set();
        // Отображение значений полей:
        obj.show();
    }
}
```

```
// Присваивание значений полям:
obj.set(200);
// Отображение значений полей:
obj.show();
// Присваивание значений полям:
obj.set(300,'C');
// Отображение значений полей:
obj.show();
}
}
```

Пример 6. Использование конструктора

```
using System;
// Описание класса с конструктором:
class MyClass{
    // Закрытые поля:
    public int num; // Целочисленное поле
    public char symb; // Символьное поле
    public string txt; // Текстовое поле
    // Открытый метод для отображения значений полей:
    public void show(){
        Console.WriteLine("Поля: {0}, '{1}' и \"{2}\"", num, symb, txt);
    }
    // Конструктор без аргументов:
    public MyClass(){
        // Значения полей:
        num=100;
        symb='A';
        txt="Красный";
    }
    // Конструктор с одним целочисленным аргументом:
    public MyClass(int n){
        // Значения полей:
        num=n;
        symb='B';
        txt="Желтый";
    }
}
```



```
// Конструктор с двумя аргументами:
public MyClass(int n, char s){
// Значения полей:
num=n;
symb=s;
txt="Зеленый";
}
// Конструктор с тремя аргументами:
public MyClass(int n, char s, string t){
// Значения полей:
num=n;
symb=s;
txt=t;
}
// Конструктор с одним текстовым аргументом:
public MyClass(string t){
// Значения полей: num=0;
symb='Z';
txt=t;
}
}
// Класс с главным методом:
class ConstructorsDemo{
// Главный метод:
static void Main(){
// Создание объектов.
// Вызывается конструктор без аргументов:
MyClass A=new MyClass();
// Проверяются значения полей объекта:
A.show();
// Вызывается конструктор с целочисленным аргументом:
MyClass B=new MyClass(200);
//Проверяются значения полей объекта:
B.show();
// Вызывается конструктор с двумя аргументами:
MyClass C=new MyClass(300,'C');
// Проверяются значения полей объекта:
```

```

C.show();
// Вызывается конструктор с тремя аргументами:
MyClass D=new MyClass(400,'D',"Синий");
// Проверяются значения полей объекта:
D.show();
// Вызывается конструктор с символьным аргументом:
MyClass F=new MyClass('A');
// Проверяются значения полей объекта:
F.show();
// Вызывается конструктор с текстовым аргументом:
MyClass G=new MyClass("Серый");
// Проверяются значения полей объекта:
G.show() ;
}
}

```

Варианты индивидуальных заданий

Индивидуальные задания взять из Лабораторной работы №4. Решить задачи используя собственные классы.

Шкала оценивания индивидуальных заданий

Примеры	1-4 балла
Примеры +Задание 1	1-6 баллов
Примеры + Задание 1 + Задание 2	1-8 баллов
Примеры + Задание 1 + Задание 2 + Выполнение одного из заданий с использованием формы	1-10 баллов

Содержание отчета:

1. Номер и тема лабораторной работы.
2. Цель лабораторной работы.
3. Техническое оснащение.
4. Скриншоты выполнения примеров
5. При выполнении индивидуальных заданий в отчет внести изображение кода программы и окно выполнения программы.
6. При выполнении программы с использованием формы, в отчет внести изображение формы с использованными элементами и модифицированной формы, так же окно программы после выполнения.
7. Вывод по лабораторной работе