Лабораторная работа по теме

«Базовые возможности языка»

1. Изучите исходный код программы, приведенной в качестве примера.
2. Объясните и продемонстрируйте отличие между свойствами Property1-Property5.
3. Объясните назначение модификатора поля Readonly. В чем его особенность.
4. Объясните назначение модификатора const. Отличие const от readonly.
5. По очереди исследуйте выполнение секций case1-case3. Что вызывается всегда в первую очередь?
6. Исследуйте класс ExampleExt. Зачем нужны и как работают методы расширения?
7. Охарактеризуйте открытые методы System.Object.
8. Охарактеризуйте закрытые методы System.Object.
9. Исследуйте секции case4-case5. Раскомментируйте соответствующие методы в классе Example.
10. Объясните назначение и варианты использования директивы using.
11. Какие типы данных используются в C#?
12. Что такое ссылочные типы? Какие типы относятся к ним?
13. Какие типы относятся к типам-значениям? В чем отличие между ссылочными и значимыми типами данных?
14. Как конвертировать значимый тип в ссылочный? Что происходит в памяти при упаковке и распаковке значимого типа? Исследуйте модуль case6.
15. Определить класс (по вариантам), в котором:
    i) конструктор без параметров;
    ii) конструктор с параметрами;
    iii) конструктор копирования;
    iv) деструктор;
    v) статический конструктор;
    vi) статический метод;
    vii) статическое поле;
    viii) свойства (классическое, автоматическое, get only, set only);
    ix) методы с использованием ref/out параметров;
    x) переопределить методы Equals, GetHashCode, ToString.
16. Создайте статический класс, содержащий методы математического преобразования над объектом вашего класса или расчета определенных параметров (уменьшение, поворот, площадь, периметр и т.п.).
17. Добавьте к созданному классу методы расширения (например: проверки возможности упаковки вашей геометрической фигуры в коробку размера a,b,c).
18. Создайте и выведите анонимный тип (по образцу вашего класса).
19. Продемонстрируйте механизм упаковки и распаковки.

Варианты заданий:

Вариант	Задание
1	Пятиугольник
2	Куб
3	Треугольная пирамида
4	Ромб
5	Цилиндр
6	Квадратная пирамида
7	Призма
8	Прямой конус
9	Усечённый конус
10	Кольцо
11	Параллелепипед
12	Трапеция
13	Треугольная призма
14	Звезда
15	Сфера

Пример программы:

```
using System;

namespace Example
{
    internal class Program
    {
        static void Main(string[] args)
        {
            //case 1
            //Example.MethodStatic(1);

            //case 2
            //Example example = new Example();
            //example.ToString();

            //case 3
            //Example.StaticProperty = 10;

            //case 4
            //Object obj = new Example();
            //Console.WriteLine(obj.GetType());

            //Object i32 = new Int32();
            //Console.WriteLine(i32.GetType());
            //i32 = 4;
            //int i = 4;
            //Console.WriteLine((i == (int)i32).ToString());

            //case 5
            //var ex1 = new Example();
            //ex1.Property4 = 10;

            //var ex2 = new Example();
            //ex2.Property4 = 10;
```

```
            //Console.WriteLine(ex1.Equals(ex2));
            //Console.WriteLine(ex1.GetHashCode());

            //case 6
            //int x = 100;
            //Object oX = x;
            //int y = (int)oX;
        }
    }

    internal class Example
    {
        private static Random _rand;
        public readonly string ReadOnlyStr;
        public const string ConstStr = "CONST STRING";

        #region Properties

        public static int StaticProperty { get; set; }

        public Int32 Property1 { get; set; }

        public string Property2 { get; }

        public double Property3 { get; private set; }

        private int property4;
        public int Property4
        {
            get { return this.property4; }
            set { this.property4 = value; }
        }

        public int Property5 { private get; set; }

        #endregion

        #region Constructors

        static Example()
        {
            Console.WriteLine("Статический констуктор");
            StaticProperty = 10;
            _rand = new Random();
        }

        public Example()
        {
            Console.WriteLine("Констурктор без параметров");

            ReadOnlyStr = "BSTU";
            Property2 = "Hello World";
            Property4 = Int32.MaxValue;
        }

        public Example(string str) : this()
        {
            Console.WriteLine("Конструктор с параметром");

            Property2 = str;
        }

        public Example(Example example)
        {
            Console.WriteLine("Конструктор копирования");
```

```
            Property1 = example.Property1;
            Property2 = example.Property2;
        }

        #endregion

        #region Methods

        public static int MethodStatic(int x)
        {
            Console.WriteLine("Статический метод");
            return x * StaticProperty;
        }

        public int MethodRef(ref int val)
        {
            Console.WriteLine("Передача параметров по ссылке: ref");
            val *= StaticProperty + _rand.Next(-100, 100);
            return val - _rand.Next();
        }

        public void MethodOut(out string str)
        {
            Console.WriteLine("Передача параметров по ссылке: out");
            str = _rand.Next().ToString();
        }

        public override String ToString()
        {
            return String.Format("({0}, {1}, {2}, {3})", Property1, Property2, Property3,
Property4, StaticProperty);
        }

        //case4-case5
        //public override bool Equals(object obj)
        //{
        //    if (obj == null || obj.GetType() != this.GetType()) return false;

        //    var ex = (Example)obj;
        //    return (Property1 == ex.Property1) && (Property2 == ex.Property2);
        //}

        //case4-case5
        //public override int GetHashCode()
        //{
        //    return Property4 ^ Property1 | StaticProperty;
        //}

        #endregion
    }

    internal static class ExampleExt
    {
        public static int MethodExt(this Example ex, int x)
        {
            return ex.Property1 + ex.Property4 + x;
        }
    }

}
```