

# Модуль 2, практическое занятие 3-4

**Классы. Члены классов  
Объекты**

```
using System;

namespace HelloApp
{
    class Person
    {
        public string name; // имя
        public int age;     // возраст

        public void GetInfo()
        {
            Console.WriteLine($"Имя: {name} Возраст: {age}");
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            Person tom = new Person();
            tom.GetInfo();      // Имя: Возраст: 0

            tom.name = "Tom";
            tom.age = 34;
            tom.GetInfo();    // Имя: Том Возраст: 34

            Console.Read();
        }
    }
}
```

Имя: Возраст: 0  
Имя: Том Возраст: 34

```
class Person
{
    public string name;
    public int age;

    public Person() { name = "Неизвестно"; age = 18; }           // 1 конструктор

    public Person(string n) { name = n; age = 18; }           // 2 конструктор

    public Person(string n, int a) { name = n; age = a; }           // 3 конструктор

    public void GetInfo()
    {
        Console.WriteLine($"Имя: {name} Возраст: {age}");
    }
}

static void Main(string[] args)
{
    Person tom = new Person();           // вызов 1-ого конструктора без параметров
    Person bob = new Person("Bob");      // вызов 2-ого конструктора с одним параметром
    Person sam = new Person("Sam", 25);  // вызов 3-его конструктора с двумя параметрами
}
```

```
bob.GetInfo();           // Имя: Bob Возраст: 18
tom.GetInfo();           // Имя: Неизвестно Возраст: 18
sam.GetInfo();           // Имя: Sam Возраст: 25
```

```
Console.ReadKey();
```

Имя: Неизвестно Возраст: 18

Имя: Bob Возраст: 18

Имя: Sam Возраст: 25

```
class Person
{
    public string name;
    public int age;

    public Person() : this("Неизвестно")
    {
    }

    public Person(string name) : this(name, 18)
    {
    }

    public Person(string name, int age)
    {
        this.name = name;
        this.age = age;
    }

    public void GetInfo()
    {
        Console.WriteLine($"Имя: {name} Возраст: {age}");
    }
}
```

# Задача 1

Программа проверяет скоро ли день рождения? (Без учета високосного года)

Получите от пользователя имя и дату рождения (год, месяц, число), создайте объект класса `Birthday`, описывающего сведения о человеке. Получите число дней до очередного дня рождения.

Используются возможности библиотечной структуры `System.DateTime`  
([https://msdn.microsoft.com/ru-ru/library/system.datetime\(v=vs.110\).aspx](https://msdn.microsoft.com/ru-ru/library/system.datetime(v=vs.110).aspx))

# Задача 1

```
class Birthday {  
    string name; // закрытое поле - фамилия  
    int year, month, day; // Закрытые поля: год, месяц, день рождения  
    public Birthday(string name, int y, int m, int d) { // Конструктор  
        this.name = name;  
        year = y; month = m; day = d;  
    }  
    DateTime Date { // закрытое свойство - дата рождения  
        get { return new DateTime(year, month, day); }  
    }  
    public string Information { // свойство - сведения о человеке  
        get {  
            return name + ", дата рождения " + day + ":" + month + ":" + year;  
        }  
    }  
    /// Свойство HOWMANYDAYS  
}
```

# Задача 1

```
public int HowManyDays { // свойство - сколько дней до дня рождения
    get {
        // номер сего дня от начала года:
        int nowDOY = DateTime.Now.DayOfYear;
        // номер дня рождения от начала года:
        int myDOY = Date.DayOfYear;
        int period = myDOY >= nowDOY ? myDOY - nowDOY :
                                         365 - nowDOY + myDOY;
        return period;
    }
}
```

# Задача 1

```
class Program {
    static void Main( ) {
        Birthday md = new Birthday("Чапаев", 1887, 2, 9);
        Console.WriteLine(md.Information);
        Console.WriteLine("До следующего дня рождения дней осталось: ");
        Console.WriteLine(md.HowManyDays);

        Birthday km = new Birthday("Маркс Карл", 1818, 5, 4);
        Console.WriteLine(km.Information);
        Console.WriteLine("До следующего дня рождения дней осталось: ");
        Console.WriteLine(km.HowManyDays);
    }
}
```

# Задание к задаче 1

1. Добавьте в класс **Birthday** конструктор без параметров, устанавливающий поля объекта класса в состояние «1 января 1970».
2. Добавьте в класс **Birthday** свойства, позволяющие получить информацию от дне рождения со следующими форматами представления даты: **DD Month YYYY**, **DD-MM-YY**.

## Задача 2

$$\Phi = \begin{cases} \arctan\left(\frac{y}{x}\right), & x > 0, y \geq 0 \\ \arctan\left(\frac{y}{x}\right) + 2\pi, & x > 0, y < 0 \\ \arctan\left(\frac{y}{x}\right) + \pi, & x < 0 \\ \frac{\pi}{2}, & x = 0, y > 0 \\ \frac{3\pi}{2}, & x = 0, y < 0 \\ 0, & x = 0, y = 0 \end{cases}$$
$$r^2 = y^2 + x^2$$

Класс "точка на плоскости" (**Point**):

- Автореализуемые вещественные свойства **X** и **Y** задают декартовы координаты точки.
- Свойства **для чтения** **r** и **ф** - полярные координаты точки.
- Конструктор общего вида и конструктор умолчания.

В основной программе создать два объекта класса **Point** (две точки), ввести данные о третьей точке и вывести сведения о трех точках в порядке возрастания их расстояний от начала координат.

Конец работы программы – ввод двух нулевых значений координат

## Задача 2

```
class Program    {
class Point {
    public double X { get; set; }
    public double Y { get; set; }
    public Point(double x, double y) { X = x; Y = y; }
    public Point() : this (0,0) { } // конструктор умолчания
// СВОЙСТВО RO
// СВОЙСТВО FI
    public string PointData {      // СВОЙСТВО
        get {
            string maket = "X = {0:F2}; Y = {1:F2}; Ro = {2:F2}; Fi = {3:F2} ";
            return string.Format(maket, X, Y, Ro, Fi);
        }
    }
}
```

## Задача 2

```
public double Ro {  
    get {  
        //#TODO1: реализовать свойство  
    }  
}  
  
public double Fi {  
    get {  
        //#TODO2: реализовать свойство  
    }  
}
```

## Задача 2

```
static void Main() {  
    Point a, b, c;  
    a = new Point(3, 4);  
    Console.WriteLine(a.PointData);  
    b = new Point(0,3);  
    Console.WriteLine(b.PointData);  
    c = new Point();  
    double x = 0, y = 0;  
    do {  
        Console.Write("x = ");  
        double.TryParse(Console.ReadLine(), out x);  
        Console.Write("y = ");  
        double.TryParse(Console.ReadLine(), out y);  
        c.X = x; c.Y = y;  
    // #TODO3: следующий слайд  
    } while (x != 0 | y != 0);  
}
```

## Задача 2

#TODO3: В основной программе создать два объекта класса (две точки), ввести данные о третьей точке и вывести сведения о трех точках в порядке возрастания их расстояний от начала координат.

# Задача 3

Определить класс **правильных** многоугольников, в котором конструктор с умалчивающими значениями аргументов играет роль конструктора умолчания. Поля класса – число сторон (целое) и радиус вписанной окружности (вещественный). Свойства – периметр и площадь многоугольника. Общедоступный метод **PolygonData()** формирует и возвращает строку со значениями полей и свойств объекта.

**В основной программе определить одну ссылку с типом класса.**

**Создать объект, используя конструктор умолчания, затем вводить характеристики многоугольника и выводить сведения о них.**

# Задание к задаче 3

1. В основной программе создайте массив объектов типа **Polygon**, количество объектов получить от пользователя.
2. Данные по каждому объекту вводит пользователь.
3. Программа определяет площади всех многоугольников и выводит данные о всех объектах. Площадь объекта с минимальной площадью выводится зелёным цветом, площадь объекта с максимальной площадью выводится красным цветом.
4. \* Модифицировать программу, убрав необходимость ввода количества объектов. Ввод данных продолжается для тех пор, пока не введён объект с нулевыми характеристиками. Информация обо всех введённых объектах выводится на экран после добавления каждого нового объекта.

# Задача 4

Класс «число и его шестнадцатеричные цифры»

```
public class HexNumber { // представление неотрицательных целых  
    uint number; // целое неотрицательное число  
    char[] hexView; // Шестнадцатеричное представление  
    public HexNumber(uint n) { // конструктор общего вида  
        number = n;  
        hexView = series(n);  
    }  
    public HexNumber() : this(0) {} // конструктор умолчания  
    // СВОЙСТВА КЛАССА: Number, HexView, Record  
    // МЕТОД series, ВОЗВРАЩАЮЩИЙ МАССИВ ШЕСТНАДЦАТЕРИЧНЫХ ЦИФР  
} // HexNumber
```

# Задача 4

```
public uint Number
{
    // Свойство: десятичное целое
    get { return number; }
    set
    {
        number = value;
        hexView = series(value);
    }
}
public char[] HexView
{
    // Свойство: массив символов-цифр
    get { return hexView; }
}
public string Record
{
    // Свойство: строковое представление (шестнадцатеричное) числа
    get
    {
        string str = new String(hexView);
        return "0x" + str;
    }
}
```

# Задача 4

```
// Возвращает массив шестнадцатеричных цифр числа-параметра.
char[] series(uint num)      {
    int arLen = num == 0 ? 1 : (int)Math.Log(num, 16) + 1;
    char[] res = new char[arLen];
    for (int i = arLen - 1; i >= 0; i--)           {
        uint temp = (uint)(num % 16);
        if (temp >= 0 & temp <= 9) res[i] = (char)('0' + temp);
        else res[i] = (char)('A' + temp % 10);
        num /= 16;
    }
    return res;
} // series
```

# Задача 4

Код метода Main()

```
HexNumber hex;           // ссылка с типом класса
hex = new HexNumber(0); // объект класса
uint number;
while (true) { // цикл для ввода разных значений числа
    do Console.WriteLine("Введите целое неотрицательное число: ");
    while (!uint.TryParse(Console.ReadLine(), out number));

    hex.Number = number;      // Изменяем объект через свойство
    Console.WriteLine("Свойство Number: " + hex.Number);
    Console.Write("Шестнадцатеричные цифры числа: ");

    foreach (char h in hex.HexView) Console.Write("{0} ", h);

    Console.WriteLine("\nШестнадцатеричная запись: " + hex.Record);
    Console.WriteLine("Для выхода нажмите клавишу ESC");

    if (Console.ReadKey(true).Key == ConsoleKey.Escape) break;
} // while
```

# Задача 5

1. Объявить класс **ConsolePlate**, представляющий символ консольного окна. Закрытые поля класса – символ **\_plateChar** и цвет символа **\_plateColor** (типа **ConsoleColor**) инициализированы.
2. В классе **ConsolePlate** два конструктора:
  - 2.1 без параметров, устанавливающий символ равным ‘+’
  - 2.2. с параметрами, определяющими символ и его цвет.
3. Определить свойства доступа к полям класса. Свойство доступа к полю **\_plateColor** для всех символов с кодами меньшими или равными **31 (1Fh)** присваивает полю **\_plateColor** значение ‘+’
4. В методе **Main()** класса **Program** создать массив объектов типа **ConsolePlate** и вывести все символы массива на экран, окрашивая в цвет, определяемый полем **\_plateColor** .

# Задание к задаче 5

1. Измените код класса **ConsolePlate** так, чтобы допустимыми были только заглавные символы латинского алфавита. Значение по умолчанию – символ **A**. Добавьте в класс **ConsolePlate** поле – цвет фона символа и свойство для доступа к нему. При установке значений полей объекта должно соблюдаться условие: цвет фона и цвет символа – различны.
2. Измените код основного приложения. Создайте два объекта типа **ConsolePlate**. Первый объект – символ **X**, белого цвета на красном фоне. Второй объект – символ **O** белого цвета на розовом фоне. Используя полученные объекты сформируйте в консольном окне клетчатое поле размером **NxN** плиток ( $2 \leq N < 35$ ). **N** – вводится пользователем с клавиатуры

# Задача 6

Создать класс со статическими членами, константами и полями (`readonly`) только для чтения. В отладочном режиме (начиная с первого оператора класса – поставьте отметку на `readonly string name =`) проследить (по F11) последовательность выполнения инициализаций и ....

# Задача 6

```
class myClassmate { // Одноклассник
    readonly string name = "Неизвестный"; // Фамилия
    readonly int birthYear = 1998; // год рождения
    const int apprenticeship = 4; // срок обучения (лет)
    static int entranceYear = 2016; // год поступления
    static myClassmate() {
        entranceYear = 2015;
    }
    public myClassmate() { } // Конструктор умолчания
    public myClassmate(string name, int by) { // Конструктор общего вида
        this.name = name;
        birthYear = by;
    }
    public string Information() { // Метод объекта
        return "Фамилия: " + name + "; возраст: " +
            (entranceYear - birthYear) +
            " лет; год окончания: " +
            (entranceYear + apprenticeship);
    }
}
```

# Задача 6

```
class Program {
    static void Main( )    {
        myClassmate Nan = new myClassmate();
        Console.WriteLine(Nan.Information());
        myClassmate Bob = new myClassmate("Смирнов", 1997);
        Console.WriteLine(Bob.Information());
    }
}
```

# Задача 7

Класс с индексатором представляет математическую функцию одного аргумента, определенную только в пределах от  $X_{\min}$  до  $X_{\max}$ . Вне этого интервала функция равна нулю.

Для определенности, пусть класс представляет отрезок функции  $\sin(x)$  на интервале  $[X_{\min}, X_{\max}]$ .

В основной программе определить объект класса для  $X_{\min}=0$ ,  $X_{\max}=\pi$  и, используя его индексатор, вычислить для отрезка функции, представляемой объектом класса, определенный интеграл методом трапеций (или прямоугольников).

# Задача 8

```
class Schedule {// Начало занятий в разные дни недели
    public string this[string day] {
        get {
            switch (day) {
                case "понедельник": return days[0] == null ? "Нет занятий" : days[0];
                case "вторник": return days[1] == null ? "Нет занятий" : days[1];
                case "среда": return days[2] == null ? "Нет занятий" : days[2];
                case "четверг": return days[3] == null ? "Нет занятий" : days[3];
                case "пятница": return days[4] == null ? "Нет занятий" : days[4];
                case "суббота": return days[5] == null ? "Нет занятий" : days[5];
                case "воскресенье": return days[6] == null ? "Нет занятий" : days[6];
                default: return "Ошибка в обращении!";
            }
        }
    }

    string[] days = new string[7]; // все null по умолчанию

    public Schedule(params string[] d) {
        for (int i = 0; i < d.Length; i++)
            days[i] = d[i];
    }
} // Schedule
```

# Задача 8

```
class Program {
    static void Main() {
        Schedule Module_2 = new Schedule("9_00", "10_30",
                                         null, "15_00", "13_40");
        Console.WriteLine("Начало занятий в модуле 2:");
        Console.WriteLine("понедельник: \t" + Module_2["понедельник"]);
        Console.WriteLine("вторник: \t" + Module_2["вторник"]);
        Console.WriteLine("среда: \t" + Module_2["среда"]);
        Console.WriteLine("четверг: \t" + Module_2["четверг"]);
        Console.WriteLine("пятница: \t" + Module_2["пятница"]);
        Console.WriteLine("суббота: \t" + Module_2["суббота"]);
        Console.WriteLine("воскресенье: \t" + Module_2["воскресенье"]);
    }
}
```

# Домашнее задание

1. Определить класс **Circle** с полем радиус **\_r** и свойством доступа к нему, значение радиуса положительное вещественное число. В классе **Circle** описать конструктор без параметров и конструктор с вещественным параметром. Определить свойство **S** – площадь круга заданного радиуса. В основной программе получить от пользователя диапазон изменения значения радиуса: (**Rmin**, **Rmax**), **Rmin**, **Rmax** – произвольные вещественные числа и величину шага **delta** разбиения данного диапазона. Создать объект типа **Circle**, последовательно изменяя значение радиуса на **delta** вычислять и выводить на экран значение площади круга, ограниченного данной окружностью.
2. Определить класс **LatinChar** с полем **\_char** и свойством доступа к нему, значение поля – символ латинского алфавита. Значение поля по умолчанию – ‘**a**’. Определить конструкторы класса. В основной программе создать объект типа **LatinChar** и, последовательно перебирая все символы из заданного пользователем диапазона [**minChar**, **maxChar**], выводить значение поля **\_char** объекта.

**5.2-0** Описать класс комплексных чисел `Complex`. Комплексные числа имеют вид  $a + bi$ , где  $i = \sqrt{-1}$ ,  $a, b \in \mathbb{R}$ . Определить в нем:

- конструктор, принимающий действительную и мнимую часть;
- копирующий конструктор;
- методы `Re` и `Im`, возвращающие мнимую и действительную части;
- методы `Abs` и `Arg`, возвращающие модуль и аргумент числа;
- операции сложения, вычитания, умножения и деления (аргументы могут быть как комплексными, так и комплексным и действительным числами);
- переопределённый унаследованный `ToString`.

Предусмотреть возможные исключительные ситуации, если это необходимо.

Написать программу, использующую этот класс.