

Класс и структура

Фоменко М.Ю.

Хламовник

- Красное – основная программа
- Желтое – логика «мудростей»
- Голубое – реверс строки
- Зеленое – предсказание погоды
- Белое – логика подсчета чихов Бубы
- Розовое – рисовалка кота в консоли
- Коричневое – конвертация огурцов в попугаев (кринж от нейронки чеснока)

Совершенно разная логика написана в одном **классе** (файле, пока что)

```
namespace CodeDumpExample
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Привет, кодовый мирок! Я - такая программа. Выберите действие:");
            Console.WriteLine("1. Окажить случайную мудрость");
            Console.WriteLine("2. Интерпретировать строку наоборот");
            Console.WriteLine("3. Предсказать погоду (шутка)");
            Console.WriteLine("4. Узнать, сколько раз чихнул Боб"); // Шутка, но совершенно лямбда логика
            Console.WriteLine("5. Нарисовать котика");
            Console.WriteLine("6. Конвертировать огурцы в попугаев"); // Полный бред для демонстрации

            string choice = Console.ReadLine();

            switch (choice)
            {
                case "1":
                    Console.WriteLine(GetRandomWisdom());
                    break;
                case "2":
                    Console.WriteLine("Введите строку: ");
                    string inputString = Console.ReadLine();
                    Console.WriteLine($"Интерпретированная строка: {ReverseString(inputString)}");
                    break;
                case "3":
                    PredictWeather(); // Эта функция вообще пустая, просто для примера
                    break;
                case "4":
                    Console.WriteLine($"Боб чихнул {BobSneezeCount()} раз. Ого!");
                    break;
                case "5":
                    DrawCat();
                    break;
                case "6":
                    Console.WriteLine("Сколько огурцов? ");
                    if (int.TryParse(Console.ReadLine(), out int cucumbers))
                    {
                        Console.WriteLine($"Огурцов = {ConvertCucumbersToParrots(cucumbers)} попугоночек!");
                    }
                    else
                    {
                        Console.WriteLine("Ты че огурцы считать не умеешь?");
                    }
                    break;
                default:
                    Console.WriteLine("Что-то не то ввел, бандит.");
                    break;
            }

            Console.WriteLine("Нажми любую кнопку, чтобы закрыть программу, или продолжить.");
            Console.ReadKey();
        }

        // --- Начало блока "почему тут вообще это лежит?" ---
        static string GetRandomWisdom()
        {
            string[] wisdoms = new string[]
            {
                "Если код не компилируется, то это не код.",
                "Два века бесконечны. Вселенная и человеческая глупость. Хотя насчет Вселенной я не уверен.",
                "Ладнайн - это точка невозврата, после которой качество становится роскошью.",
                "Не работавши - не лоджам. Не лоджам - не учимся. Парадокс.",
                "Оптимизация без профилирования - это как менять мотор в машине, не зная, почему она не едет."
            };

            Random rand = new Random();
            return wisdoms[rand.Next(wisdoms.Length)];
        }

        static string ReverseString(string s)
        {
            char[] charArray = s.ToCharArray();
            Array.Reverse(charArray);
            return new string(charArray);
        }

        static void PredictWeather()
        {
            Console.WriteLine("Погода предсказана! Но я тебе не скажу. Секрет."); // Туда шутка, но по факту мусор
        }

        static int BobSneezeCount()
        {
            // Эта функция тут вообще не к делу и к гораду
            // Просто какая-то внутренняя "бесмысленная" логика, которая попала в основной файл
            Random r = new Random();
            return r.Next(1, 100);
        }

        static void DrawCat()
        {
            Console.WriteLine("g");
            // ...
        }

        static double ConvertCucumbersToParrots(int cucumbers)
        {
            // Абсолютно бессмысленная, но "забавная" функция
            // Просто чтобы показать, что тут может быть любая логика
            const double PARROTS_PER_CUCUMBER = 0.73; // Не спрашивай почему
            return cucumbers * PARROTS_PER_CUCUMBER;
        }

        // --- Конец блока "почему тут вообще это лежит?" ---
    }
}
```

Организация кода

- Функции (уже были)
- Паттерны и проектирование (будут потом)

Организация по файлам

- Классы – `class`
- Структура – `struct`

Что такое класс

- Class - тип данных, созданный пользователем. Класс содержит данные (поля) и код, который управляет этими данными (функции).
- Класс это шаблон, по которому определяется форма объекта, но сам класс — абстракция: его объектное представление появится в оперативной памяти лишь после того, как будет создан объект этого класса.
- Класс – ссылочный тип данных
- **Класс и объект класса**

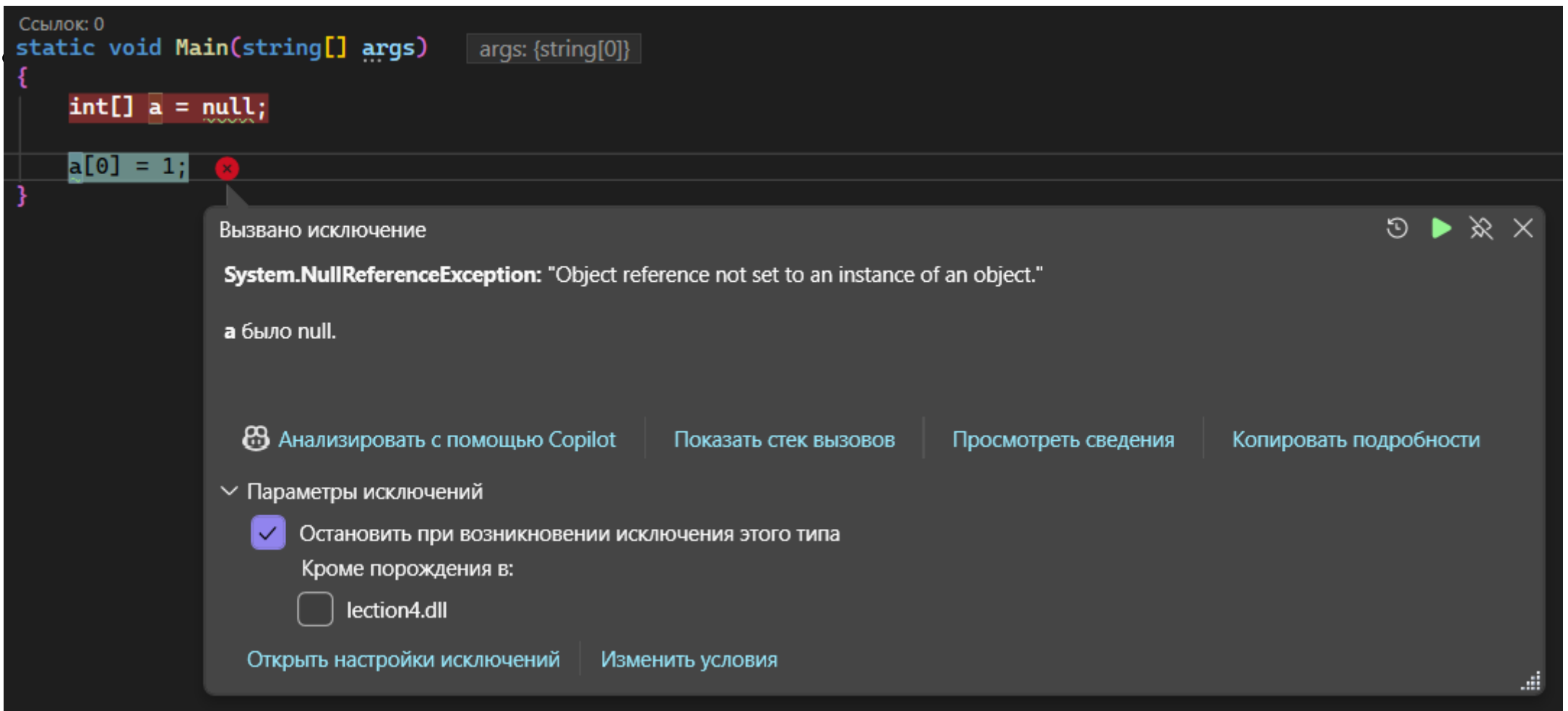
Что такое класс

- Class - тип данных, созданный пользователем. Класс содержит данные (поля) и код, который управляет этими данными (функции).
- Класс это шаблон, по которому определяется форма объекта, но сам класс — абстракция: его объектное представление появится в оперативной памяти лишь после того, как будет создан объект этого класса.
- Класс – ссылочный тип данных
- **Класс и объект класса**

null

- null - означает «**нет объекта**». Это не встроенная константа, а значение, которое может быть присвоено переменной-указателю.

null



static

- Ключевое слово **static** в языке программирования C# применяется к переменным и методам, делая их принадлежащими не объектам, а самому типу.
- static-элементы существуют независимо от создания экземпляров класса и доступны для использования даже без их создания.

Что такое структура

- Структура – составной тип данных, который объединяет несколько переменных, часто различных типов, под одним именем. В структуре могут быть собраны различные объекты: переменные, массивы, указатели, другие структуры.
- Объявление структуры создаёт шаблон, который можно использовать для создания её объектов (экземпляров).
- Структура – значимый тип данных

А зачем?

- С помощью классов мы группируем связанную логику.
- Код становится чище и легче для чтения/поиска.
- Ваши фильтры не засоряют глобальное пространство имен.

Sample class

Методы из BasicMath мы можем использовать написав название класса и название нужного нам метода.

Для использования методов из CommonMath мы должны создать объект класса этой математики (переменная) и в ней мы используем метод.

При работе в консольных приложениях мы всегда делаем классы и методы статическими

```
Ссылка: 0
internal class Program
{
    Ссылка: 0
    static void Main(string[] args)
    {
        Console.WriteLine(BasicMath.Sum(5, 6));

        CommonMath math = new CommonMath();
        Console.WriteLine(CommonMath.Pi);
        Console.WriteLine(math.someVar);
        Console.WriteLine(math.Pow(5, 6));
    }
}

Ссылка: 1
static class BasicMath
{
    Ссылка: 1
    static public int Sum(int x, int y)
    {
        return x + y;
    }

    Ссылка: 0
    static public int Minus(int x, int y)
    {
        return x - y;
    }
}

Ссылка: 3
class CommonMath
{
    public const double Pi = 3.14;
    public int someVar = 5;
    Ссылка: 1
    public int Pow(int x, int y)
    {
        return (int)Math.Pow(x, y);
    }
}
```

Sample struct

Структура тоже может в методы и
тоже может быть статичной

```
Ссылка: 0
internal class Program
{
    Ссылка: 0
    static void Main(string[] args)
    {
        Date birthDay = new Date();

        birthDay.day = 15;
        birthDay.year = 2007;
        birthDay.month = 9;

        Console.WriteLine(birthDay.GetString());
        //15.9.2007
    }
}

Ссылка: 2
struct Date
{
    public int year;
    public int month;
    public int day;

    Ссылка: 1
    public string GetString()
    {
        return $"{day}.{month}.{year}";
    }
}
```

Спойлеры к 4 лабе

Про как организовывать код в логические модули (static class), понять разницу и назначение struct и class, и закрепить на практике работу с передачей данных по ссылке (ref *(да опять реф)*)).

- **Класс Image:** Умеет загружать и сохранять картинки, знает свою ширину/высоту и дает доступ к пикселям по координатам image[x, y].
- **Структура Pixel:** Хранит три байта R, G, B.

Где лежат данные (куча стек)

Pixel:

- Наша структура Pixel хранит всего три байта: R, G, B. Это маленькая порция данных.
- int, double, bool. Это всё типы-значения.

Структура расположена в стеке, копирование и передача в функцию аналогично передаче типов-значений (int, bool и тд) – данные копируются.

Где лежат данные (куча стек)

Pixel:

```
Pixel p1 = new Pixel { R = 10, G = 20, B = 30 };  
Pixel p2 = p1; // p2 – это НОВАЯ, независимая копия p1  
p2.R = 99;      // p1.R по-прежнему 10. p1 НЕ ИЗМЕНИЛСЯ!
```

Плюсы: Быстро, предсказуемо, нет «неожиданных» изменений из других мест.

Минусы: Если struct большой (много полей), копирование может быть медленным.

Где лежат данные (куча стек)

Class:

Объект класса расположена в куче, ссылка хранится на стеке (полная аналогия с массивом)

- Когда вы присваиваете class одной переменной другой или передаете в метод, копируется только «адрес», где лежит оригинал.

Где лежат данные (куча стек)

Class:

```
Ссылка: 0
static void Main(string[] args)
{
    CommonMath math_1 = new CommonMath() { someVar = 104 };
    CommonMath math_2 = math_1; //math2 - это НЕ копия,
                                //это просто ЕЩЕ ОДНА ССЫЛКА на ТОТ ЖЕ объект,
                                //что и math_1

    math_2.someVar = 999; // math_1.someVar теперь тоже 999
}
```

Где лежат данные (куча стек)

Class:

Плюсы: Экономия памяти (копируем только адрес), можно легко менять один объект из разных мест.

Минусы: "Неожиданные" изменения, сборщик мусора (GC) может создавать задержки.

Мини проблема

- Пишем метод `void ChangePixel(Pixel p) { p.R = 0; }`.
- Вызываем его: `Pixel myPixel = ...; ChangePixel(myPixel);`

Что тут не так?

А) в `myPixel` красный цвет станет 0 – правая рука вверх

Б) в `myPixel` красный цвет не изменится – левая рука вверх

Мини проблема

- Пишем метод `void ChangePixel(Pixel p) { p.R = 0; }`.
- Вызываем его: `Pixel myPixel = ...; ChangePixel(myPixel);`

`myPixel` передается в `ChangePixel` по значению, то есть создается КОПИЯ. Метод `ChangePixel` меняет копию, а твой `myPixel` остается прежним.

Спасибо за внимание

- Хотите еще раз послушать про ref? Мб не понятно все еще или как.
- Можно сессию лайв кода устроить еще.