

Санкт-Петербургский Национальный Исследовательский
Университет Информационных Технологий, Механики и Оптики

Факультет инфокоммуникационных технологий

Лабораторная работа №6
Создание и использование классов

Выполнил
Стафеев И.А.

Группа
К3221

Проверил
Иванов С.Е.

Санкт-Петербург,
2024

СОДЕРЖАНИЕ

	Стр.
ВВЕДЕНИЕ	3
1 Упражнение 1.....	4
2 Упражнение 2.....	7
3 Упражнение 3.....	10
ЗАКЛЮЧЕНИЕ	14

ВВЕДЕНИЕ

Цель работы: изучение понятия класса как пользовательского типа данных и приобретение навыков работы с классами.

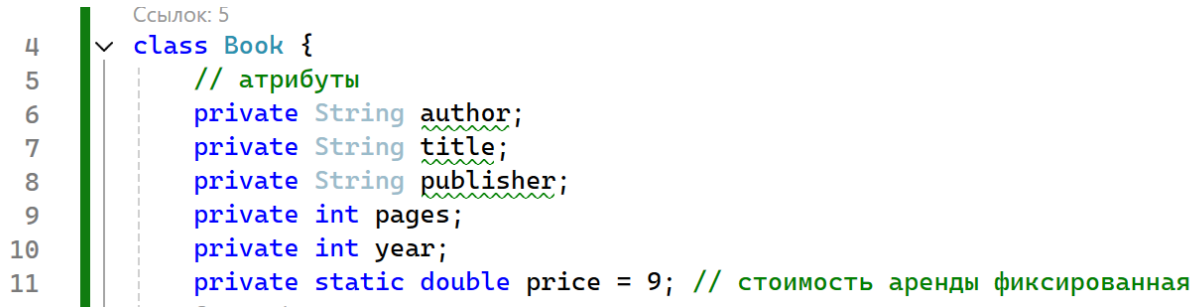
Для достижения цели необходимо выполнить следующие упражнения:

1. Разработка класса Book
2. Использование конструкторов
3. Реализация класса Triangle

1 Упражнение 1

Задача: В этом упражнении вы создадите класс Book с соответствующими полями (автор или авторы, название, год издания и стоимость аренды за книгу) и методами.

Создание класса Book и его атрибутов показано на рисунке 1.



```
Ссылка: 5
4  class Book {
5      // атрибуты
6      private String author;
7      private String title;
8      private String publisher;
9      private int pages;
10     private int year;
11     private static double price = 9; // стоимость аренды фиксированная
```

Рисунок 1 — Атрибуты класса Book

Методы для установки значений атрибутов объекта класса, установки статического поля цены аренды и подсчета стоимости аренды показаны на рисунке 2.



```
Ссылка: 1
12 public void SetBook(String author, String title, String publisher, int pages, int year)
13 { // установка атрибутов
14     this.author = author;
15     this.title = title;
16     this.publisher = publisher;
17     this.pages = pages;
18     this.year = year;
19 }
Ссылка: 1
20 public static void SetPrice(double price)
21 { // установка стоимости аренды
22     Book.price = price;
23 }
Ссылка: 1
24 public double PriceBook(int s)
25 { // стоимость аренды на s суток
26     return s * price;
27 }
```

Рисунок 2 — Методы SetBook, SetPrice и PriceBook

Метод для вывода информации о книге представлен на рисунке 3.

```

28      Ссылка: 1
29      public void Show()
30      { // вывод информации о книге
31          Console.WriteLine("Книга:");
32          Console.WriteLine("Название: {0}", title);
33          Console.WriteLine("Автор: {0}", author);
34          Console.WriteLine("Год издания: {0}", year);
35          Console.WriteLine("Количество страниц: {0}", pages);
36          Console.WriteLine("Стоимость аренды: {0}", Book.price);
37      }
38  }

```

Рисунок 3 — Метод Show

Для теста класса в методе Main был создан его экземпляр, был вызван метод заполнения атрибутов и выведена информация о книге (4).

```

40      Ссылка: 0
41      class Program {
42          Ссылка: 0
43          public static void Main(string[] args) {
44              Book b = new Book();
45              b.SetBook("У. Берроуз", "Голый завтрак", "АСТ", 320, 2019);
46              Book.SetPrice(35);
47              b.Show();
48              int n = 5;
49              Console.WriteLine("Стоимость аренды за {0} суток составит {1} у.е.", n, b.PriceBook(n));
50          }
51      }

```

Рисунок 4 — Метод Main

Результат выполнения представлена на рисунке 5.

```

Консоль отладки Microsoft Visual Studio
Книга:
Название: Голый завтрак
Автор: У. Берроуз
Год издания: 2019
Количество страниц: 320
Стоимость аренды: 35
Стоимость аренды за 5 суток составит 175 у.е.

C:\Users\Thunderobot\source\repos\Book\Book\bin\Debug\net8.0\Book.exe (
процесс 7448) завершил работу с кодом 0 (0x0).
Нажмите любую клавишу, чтобы закрыть это окно:

```

Рисунок 5 — Результат выполнения программы

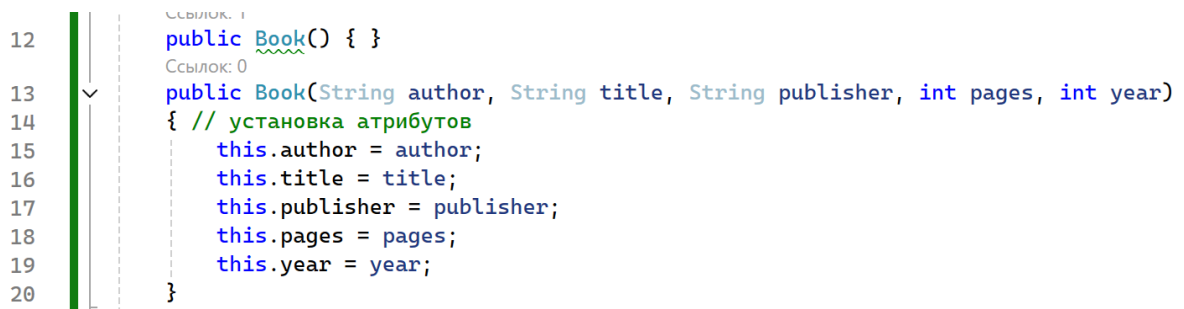
Сравнение статического метода и метода класса. Статический метод объявляется с помощью ключевого слова **static** и привязан к классу, а не к конкретному объекту класса. Он не имеет доступа к полям и методам экземпляра класса, но имеет доступ к статическим полям и статическим методам класса. Вызывается через имя класса, создавать объект класса не требуется. Используется, когда данные о конкретных объектах не нужны.

Метод класса привязан к конкретному объекту класса и имеет доступ как к полям экземпляра, так и к статическим полям и методам. Для вызова необходимо создавать экземпляр класса. Используется, когда требуется доступ к состоянию конкретного объекта класса.

2 Упражнение 2

Задача: В этом упражнении вы добавите конструкторы для инициализации объекта. Конструктор экземпляра вызывается автоматически при создании объекта класса с помощью операции `new`, причем имя конструктора совпадает с именем класса. Также вы будете использовать статический конструктор для инициализации статического поля класса.

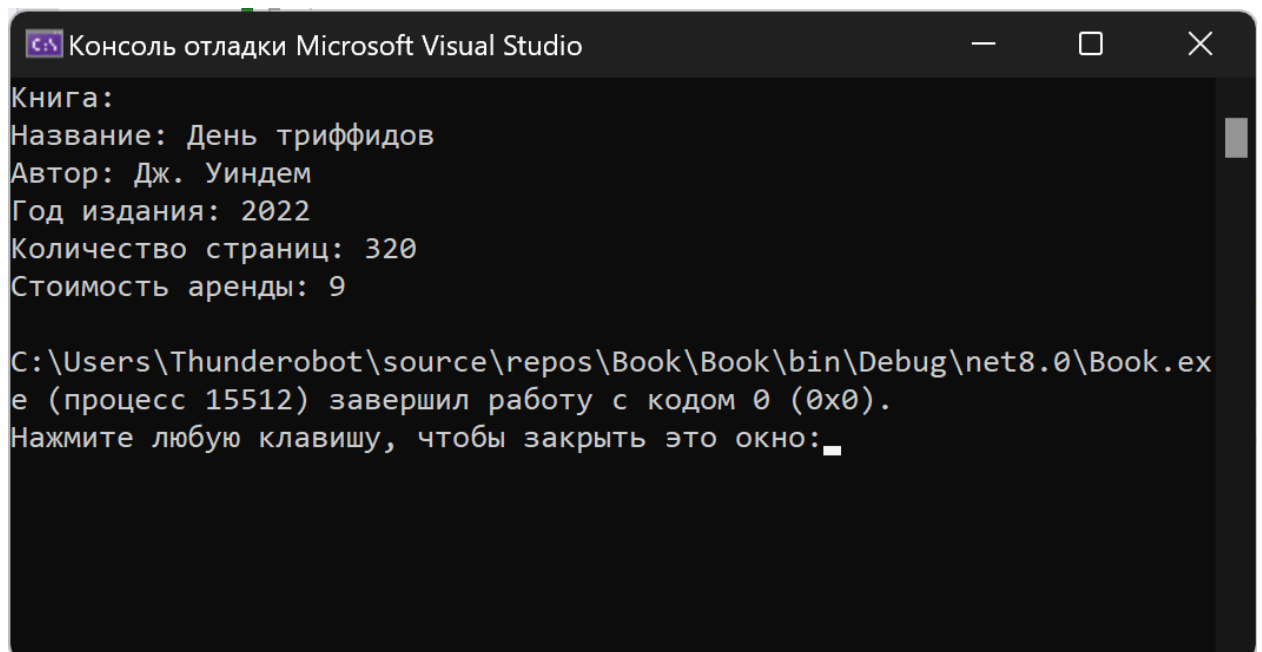
На основе созданного класса `Book` в упражнении 1 был создан новый класс `Book`. Добавлен конструктор и конструктор по умолчанию (6).



```
12  public Book() { }
13  public Book(String author, String title, String publisher, int pages, int year)
14  { // установка атрибутов
15      this.author = author;
16      this.title = title;
17      this.publisher = publisher;
18      this.pages = pages;
19      this.year = year;
20  }
```

Рисунок 6 — Конструктор класса `Book`

Пример выполнения программы с созданием экземпляра книги и выводом информации приведен на рисунке 7.



```
Консоль отладки Microsoft Visual Studio

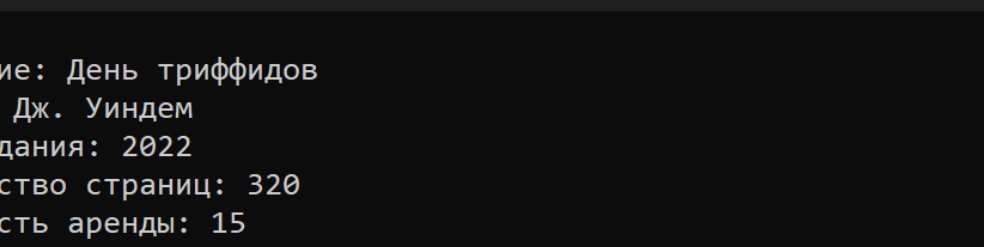
Книга:
Название: День триффидов
Автор: Дж. Уиндем
Год издания: 2022
Количество страниц: 320
Стоимость аренды: 9

C:\Users\Thunderobot\source\repos\Book\Book\bin\Debug\net8.0\Book.exe (процесс 15512) завершил работу с кодом 0 (0x0).
Нажмите любую клавишу, чтобы закрыть это окно: █
```

Рисунок 7 — Пример выполнения программы

Затем был добавлен статический конструктор, который вызывается до первого обращения к экземпляру класса и выполняет некоторые предварительные действия по инициализации (8).

Рисунок 8 — Код статического конструктора



```
Консоль отладки Microsoft Visual Studio

Книга:
Название: День триффидов
Автор: Дж. Уиндем
Год издания: 2022
Количество страниц: 320
Стоимость аренды: 15

C:\Users\Thunderobot\source\repos\Book\Book\bin\Debug\net8.0\Book.exe
(процесс 12140) завершил работу с кодом 0 (0x0).
Нажмите любую клавишу, чтобы закрыть это окно: _
```

Еще была сделана перегрузка конструктора, в которой параметрами являются только имя автора и название книги 10.

3 Упражнение 3

Задача: В этом упражнении требуется создать класс Triangle, разработав следующие элементы класса:

- Поля: стороны треугольника
- Конструктор, позволяющий создать экземпляр класса с заданными длинами сторон
- Методы, позволяющие
 1. вывести длины сторон треугольника на экран;
 2. рассчитать периметр треугольника;
 3. рассчитать площадь треугольника;
 4. реализовать проверку, позволяющую установить, существует ли треугольник с данными длинами сторон.

Был создан класс Triangle с полями - сторонами треугольника (12).

```
3 class Triangle
4 {
5     private double a, b, c;
```

Рисунок 12 — Поля класса Triangle

Для проверки на треугольник создан метод статический IsTriangle, возвращающий булево значение (13).

```
44 public static bool IsTriangle(double a, double b, double c) {
45     // проверка на треугольник
46     if (a < b + c && b < a + c && c < a + b) { return true; }
47     return false;
48 }
49 }
50 }
```

Рисунок 13 — Метод IsTriangle

Конструкторы класса показаны на рисунке 14. Для равностороннего треугольника была создана перегрузка конструктора - метод, принимающий

одну сторону. Конструктор выполняет проверку на то, что треугольник существует, и в противном случае вызывает исключение.

```
6      | Ссылка: 0
      | public Triangle() { }
      | Ссылка: 3
7      | public Triangle(double a, double b, double c)
8      | {
9      |     // конструктор
10     |     if (!Triangle.IsTriangle(a, b, c)) throw new ArgumentException();
11     |     this.a = a; this.b = b; this.c = c;
12     | }
      | Ссылка: 0
13     | public Triangle(double a)
14     | {
15     |     // перегрузка: равносторонний треугольник
16     |     if (a <= 0) throw new ArgumentException();
17     |     this.a = a; this.b = a; this.c = a;
18     | }
      | Ссылка: 3
```

Рисунок 14 — Конструктор класса Triangle

Методы для вычисления периметра и площади показаны на рисунке 15. второй метод внутри себя вызывает первый.

```
34     | Ссылка: 2
35     | public double Perimeter()
36     | {
37     |     // получение периметра
38     |     return this.a + this.b + this.c;
      | }
      | Ссылка: 1
39     | public double Square() {
40     |     // получение площади
41     |     double p = Perimeter() / 2;
42     |     return Math.Sqrt(p * (p - this.a) * (p - this.b) * (p - this.c));
43     | }
      | Ссылка: 1
```

Рисунок 15 — Методы Perimeter и Square

Метод для вывода информации о треугольнике показан на рисунке 16.

```

19  ~
20  ~
21  ~
22  ~
23  ~
24  ~
25  ~
26  ~
27  ~
28  ~
29  ~
30  ~
31  ~
32  ~
33  ~
Ссылка: 3
public void Show()
{
    // вывод информации о треугольнике
    if (this.a == this.b && this.b == this.c)
    {
        Console.WriteLine("Сторона равностороннего треугольника равна {0}", this.a);
    }
    else
    {
        Console.WriteLine("Стороны треугольника равны {0}, {1} и {2}", this.a, this.b, this.c);
    }
    Console.WriteLine("Периметр: {0}", Perimeter());
    Console.WriteLine("Площадь: {0:f4}", Square());
    Console.WriteLine();
}

```

Рисунок 16 — Метод Show

Метод Main представлен на рисунке 17. Внутри создаются экземпляры класса Triangle - разносторонний, равносторонний и несуществующий треугольник - и выводит информация о них.

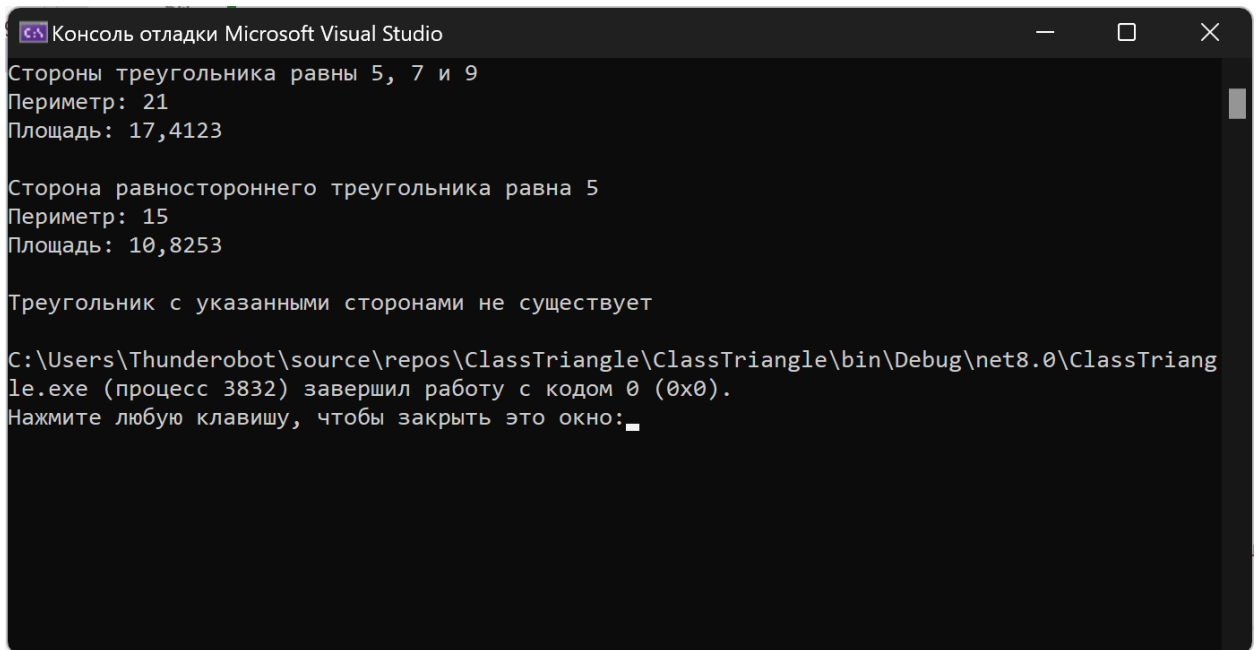
```

--
51  ~
52  ~
53  ~
54  ~
55  ~
56  ~
57  ~
58  ~
59  ~
60  ~
61  ~
62  ~
63  ~
64  ~
65  ~
66  ~
67  ~
68  ~
69  ~
Ссылка: 0
class Program {
    Ссылка: 0
    public static void Main(string[] args) {
        // разносторонний треугольник
        Triangle t1 = new Triangle(5, 7, 9);
        t1.Show();
        // равносторонний треугольник
        Triangle t2 = new Triangle(5, 5, 5);
        t2.Show();
        // не треугольник
        try
        {
            Triangle t3 = new Triangle(3, 5, 8);
            t3.Show();
        }
        catch (ArgumentException e) {
            Console.WriteLine("Треугольник с указанными сторонами не существует");
        }
    }
}

```

Рисунок 17 — Метод Main

Пример выполнения программы показан на рисунке 18.

The image shows a screenshot of the 'Консоль отладки Microsoft Visual Studio' (Microsoft Visual Studio Debug Console) window. The window has a dark background and a light-colored title bar. The output text is as follows:

```
Стороны треугольника равны 5, 7 и 9  
Периметр: 21  
Площадь: 17,4123  
  
Сторона равностороннего треугольника равна 5  
Периметр: 15  
Площадь: 10,8253  
  
Треугольник с указанными сторонами не существует  
  
C:\Users\Thunderobot\source\repos\ClassTriangle\ClassTriangle\bin\Debug\net8.0\ClassTriangle.exe (процесс 3832) завершил работу с кодом 0 (0x0).  
Нажмите любую клавишу, чтобы закрыть это окно: █
```

Рисунок 18 — Пример выполнения программы

ЗАКЛЮЧЕНИЕ

В ходе выполнения лабораторной работы были выполнены все требуемые упражнения. Цель работы достигнута. Получены знания о классе как пользовательском типе данных и приобретены навыки работы с классами.