

Санкт-Петербургский национальный исследовательский университет
информационных технологий, механики и оптики

Факультет инфокоммуникационных технологий

Направление подготовки 11.03.02

Лабораторная работа №6

Создание и использование классов

Выполнил:

Доценников Никита Андреевич

Группа: К3221

Проверил:

Иванов Сергей Евгеньевич

Санкт-Петербург

2025

Цель работы:

Изучить понятия класса как пользовательского типа данных и приобрести навыки работы с классами.

Упражнение 1. Разработка класса Book.

В этом упражнении я создал класс `Book` с полями автор, название, год издания и стоимость аренды за книгу, а также методами.

```
using System;

class Book
{
    private String author;
    private String title;
    private String publisher;
    private int pages;
    private int year;

    private static double price = 9;

    static Book()
    {
        price = 10;
    }

    public Book()
    {
    }

    public Book(String author, String title, String publisher,
int pages, int year)
    {
        this.author = author;
        this.title = title;
        this.publisher = publisher;
        this.pages = pages;
        this.year = year;
    }

    public Book(String author, String title)
```

```

{
    this.author = author;
    this.title = title;
}

public void SetBook(String author, String title, String
publisher, int pages, int year)
{
    this.author = author;
    this.title = title;
    this.publisher = publisher;
    this.pages = pages;
    this.year = year;
}

public static void SetPrice(double price)
{
    Book.price = price;
}

public void Show()
{
    Console.WriteLine("\nКнига:\n Автор: {0}\n Название:
{1}\n Год издания: {2}\n {3} стр.\n Стоимость аренды: {4}",
author, title, year, pages, Book.price);
}

public double PriceBook(int s)
{
    double cost = s * price;
    return cost;
}
}

```

Тесты:

```

public static void Main(string[] args)
{
    Book b1 = new Book();
    b1.SetBook("Пушкин А.С.", "Капитанская дочка", "Вильямс",
123, 2012);
}

```

```
Book.SetPrice(12);  
b1.Show();  
Console.WriteLine("\n Итоговая стоимость аренды: {0} р.",  
b1.PriceBook(3));  
}
```

Пример:

```
Книга:  
Автор: Пушкин А.С.  
Название: Капитанская дочка  
Год издания: 2012  
123 стр.  
Стоимость аренды: 12  
  
Итоговая стоимость аренды: 36 р.
```

Упражнение 2. Использование конструкторов.

В этом упражнении я добавил конструктор для инициализации объекта.

Тесты:

```
public static void Main(string[] args)  
{  
    Book b1 = new Book();  
    b1.SetBook("Пушкин А.С.", "Капитанская дочка", "Вильямс",  
123, 2012);  
    Book.SetPrice(12);  
    b1.Show();  
    Console.WriteLine("\n Итоговая стоимость аренды: {0} р.",  
b1.PriceBook(3));  
}
```

```

        Book b2 = new Book("Толстой Л.Н.", "Война и мир", "Наука и
жизнь", 1234, 2013);
        b2.Show();

        Book b3 = new Book("Лермонтов М.Ю.", "Мцыри");
        b3.Show();
    }

```

Примеры:

```

A labs/lab6/Book | main | dotnet run
dotnet .NET 09:32

Книга:
Автор: Пушкин А.С.
Название: Капитанская дочка
Год издания: 2012
123 стр.
Стоимость аренды: 12

Итоговая стоимость аренды: 36 р.

Книга:
Автор: Толстой Л.Н.
Название: Война и мир
Год издания: 2013
1234 стр.
Стоимость аренды: 12

Книга:
Автор: Лермонтов М.Ю.
Название: Мцыри
Год издания: 0
0 стр.
Стоимость аренды: 12

```

Упражнение 3. Реализация класса Triangle.

В этом упражнении я создал класс Triangle и разработал следующие элементы класса:

- Поля: стороны треугольника.
- Конструктор, позволяющий создать экземпляр класса с заданными длинами сторон.

- Методы, позволяющие вывести длины сторон треугольника на экран, рассчитать периметр треугольника, рассчитать площадь треугольника, реализовать проверку, позволяющую установить, существует ли треугольник с данными длинами сторон.

```
using System;

class Triangle
{
    private int a;
    private int b;
    private int c;

    public Triangle(int a, int b, int c)
    {
        this.a = a;
        this.b = b;
        this.c = c;
    }

    public void Show()
    {
        Console.WriteLine("a: {0}\nb: {1}\nc: {2}", a, b, c);
    }

    public int Perimeter()
    {
        return a + b + c;
    }

    public double Area()
    {
        double p = Perimeter() / 2;
        return Math.Sqrt(p * (p - a) * (p - b) * (p - c));
    }

    public bool Exists()
    {
        if (a <= 0 || b <= 0 || c <= 0) return false;

        double maxSide = Math.Max(Math.Max(a, b), Math.Max(b,
c));
    }
}
```

```

        if (maxSide >= a + b + c - maxSide) return false;

        return true;
    }
}

```

Тесты:

```

public static void Main(string[] args)
{
    Triangle t1 = new Triangle(3, 4, 5);
    t1.Show();
    Console.WriteLine("perimeter = {0}", t1.Perimeter());
    Console.WriteLine("area = {0}", t1.Area());
    Console.WriteLine("does this triangle exist? {0}",
t1.Exists());

    Triangle t2 = new Triangle(3, 4, 7);
    t2.Show();
    Console.WriteLine("does this triangle exist? {0}",
t2.Exists());
}

```

Пример:

```

a: 3
b: 4
c: 5
perimeter = 12
area = 6
does this triangle exist? True
a: 3
b: 4
c: 7
does this triangle exist? False

```

Выводы.

В ходе проделанной работы я изучил понятия класса как пользовательского типа данных и приобрел навыки работы с классами.

Code review. (by zzzcode.ai)

Резюме

Код, представленный выше, реализует два класса: `Book` и `Triangle`. Класс `Book` управляет информацией о книгах, включая автора, название, издателя, количество страниц и год издания. Класс `Triangle` отвечает за свойства треугольника, такие как стороны, периметр и площадь. В целом, код демонстрирует базовые принципы объектно-ориентированного программирования, но требует улучшений в области стиля, структуры и обработки ошибок.

Ошибка

В классе `Book` метод `SetBook` не проверяет входные параметры на корректность. Например, количество страниц и год издания могут быть отрицательными значениями. Это может привести к некорректным данным. Рекомендуется добавить валидацию входных данных.

Стиль кода

Код написан в целом в соответствии с общепринятыми стандартами C#. Однако, использование типа `String` вместо `string` не является стандартным стилем C#. Рекомендуется использовать `string` для повышения читаемости и соответствия стилю C#.

Структура кода

Структура классов в целом логична, но методы и свойства можно было бы организовать более последовательно. Например, методы, связанные с установкой и получением значений, можно сгруппировать вместе. Это улучшит читаемость и упростит навигацию по коду.

Читаемость

Читаемость кода можно улучшить, добавив комментарии к методам и классам, чтобы объяснить их назначение. Также стоит использовать более описательные имена переменных и методов, чтобы сделать код более самодокументируемым.

Производительность

Код не содержит явных проблем с производительностью, однако использование статического поля `price` может быть улучшено. Если цена книги должна быть уникальной для каждого экземпляра, лучше сделать это полем экземпляра.

Масштабируемость

Код может быть сложно масштабировать из-за жесткой привязки к конкретным полям и методам. Рекомендуется рассмотреть возможность использования интерфейсов или абстрактных классов для улучшения гибкости и расширяемости.

Безопасность

Код не содержит явных уязвимостей, однако отсутствие валидации входных данных может привести к проблемам. Рекомендуется добавить проверки на корректность данных, чтобы избежать потенциальных ошибок.

Обработка ошибок

Обработка ошибок в коде отсутствует. Рекомендуется использовать исключения для обработки некорректных данных и других возможных ошибок, чтобы обеспечить более надежное выполнение программы.

Заключение

Код представляет собой хорошую основу для работы с книгами и треугольниками, но требует улучшений в области стиля, структуры и обработки ошибок. Внедрение предложенных изменений повысит качество кода и упростит его поддержку в будущем.