

Министерство образования Республики Беларусь
Учреждение образования
“Брестский государственный технический университет”
Кафедра ИИТ

Лабораторная работа №3

По дисциплине: "СПП"

Выполнил:
Студент 3 курса
Группы ПО-3
Луц М. Г.
Проверил:
Крощенко А. А.

Брест 2019

Цель работы: научиться создавать и использовать классы в программах на языке программирования C#.

Вариант 14

Задание №1:

Реализовать простой класс.

Требования к выполнению

- Реализовать пользовательский класс по варианту.
- Создать другой класс с методом main, в котором будут находиться примеры использования пользовательского класса.

Для каждого класса

- Создать поля классов
- Создать методы классов
- Добавьте необходимые get и set методы (по необходимости)
- Укажите соответствующие модификаторы видимости
- Добавьте конструкторы
- Переопределить методы toString() и equals ()

Прямоугольник, заданный длинами двух сторон – Предусмотреть возможность определения площади и периметра, а так же логические методы, определяющие, является ли прямоугольник квадратом и существует ли такой прямоугольник. Конструктор должен позволить создавать объекты с начальной инициализацией. Реализовать метод equals, выполняющий сравнение объектов данного типа.

Код программы:

```
using System;

namespace lab3_1
{
    class Program
    {
        static void Main(string[] args)
        {
            RectExample();
        }
    }
}
```

```

private static void RectExample()
{
    Rectangle first = new Rectangle();
    Rectangle second = new Rectangle(3, 4);
    Console.WriteLine(first.IsExist());
    Console.WriteLine(second.Perimeter());
    first.Height = 4;
    first.Width = 4;
    Console.WriteLine(first.IsSquare());
    Console.WriteLine(second.Area());
    Console.WriteLine(first.Equals(second));
    second.Width = 4;
    Console.WriteLine(first.Equals(second));
    Console.WriteLine(second.Equals(first));
    Console.WriteLine(first);
}
}

class Rectangle
{
    private float _width;
    private float _height;
    public float Width { get => _width; set => _width = value; }
    public float Height { get => _height; set => _height = value; }
    public Rectangle() { }
    public Rectangle(float width, float height)
    {
        _width = width;
        _height = height;
    }
    public float Perimeter() => IsExist() ? Width * 2 + Height * 2 : 0;
    public float Area() => IsExist() ? Width * Height : 0;
    public bool IsSquare() => Width == Height && IsExist();
    public bool IsExist() => Width > 0 && Height > 0;
    public override bool Equals(object obj)
    {

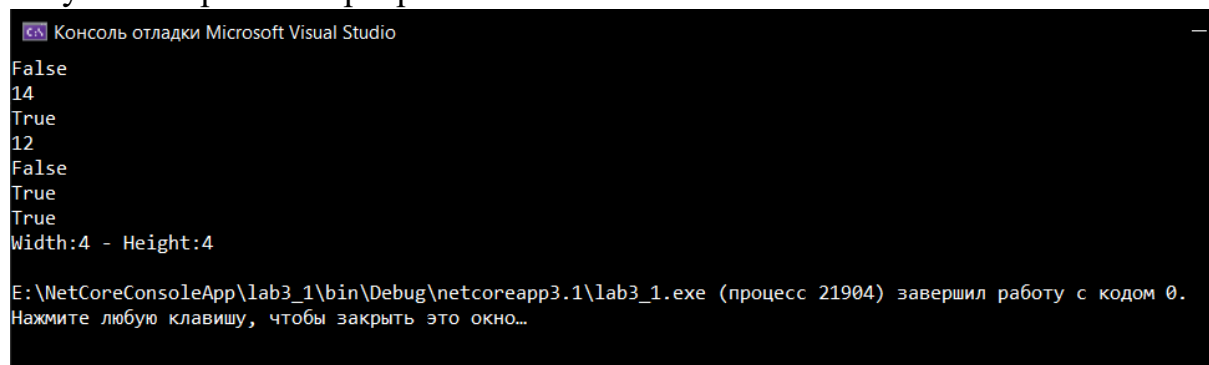
```

```

        Rectangle other = obj as Rectangle;
        if (other == null)
        {
            return false;
        }
        if (other.GetType() == GetType())
        {
            return Width == other.Width && Height == other.Height;
        }
        else
        {
            return false;
        }
    }
    public override int GetHashCode()
    {
        return GetHashCode.Combine(Width, Height);
    }
    public override string ToString()
    {
        return $"Width:{Width} - Height:{Height}";
    }
}
}

```

Результаты работы программы:



```

Консоль отладки Microsoft Visual Studio
False
14
True
12
False
True
True
Width:4 - Height:4

E:\NetCoreConsoleApp\lab3_1\bin\Debug\netcoreapp3.1\lab3_1.exe (процесс 21904) завершил работу с кодом 0.
Нажмите любую клавишу, чтобы закрыть это окно...

```

Задание №2:

Разработать автоматизированную систему на основе некоторой структуры данных, манипулирующей объектами пользовательского класса.

Реализовать требуемые функции обработки данных

Требования к выполнению

- Задание посвящено написанию классов, решающих определенную задачу автоматизации;
- Данные для программы загружаются из файла (формат произволен).
Файл создать и написать вручную.

Автоматизированная система в библиотеке:

Составить программу, которая содержит текущую информацию о книгах в библиотеке. Сведения о книгах (Book) содержат:

- номер УДК;
- Фамилию и инициалы автора;
- Название;
- Год издания;
- Количество экземпляров в библиотеке;
- Количество страниц;
- Количество томов;
- ФИО читателя, взявшего книгу (при наличии);
- Срок сдачи книги (если была взята).

Программа должна обеспечивать:

- Формирование общего списка книг;
- Формирование списка книг, старше n лет;
- Формирование списка книг, взятых на чтение;
- Формирование списка книг, взятых на чтение с выводом личной информации о читателях;
- Формирование списка книг, которые задержаны читателем дольше указанного срока.

Код программы:

```
using System;
```

```

using System.IO;
using System.Linq;
using System.Text;
using System.Collections.Generic;
using System.Xml.Serialization;
namespace lab3_2
{
    class Program
    {
        static void Main(string[] args)
        {
            Library library = new Library("books.xml");

            Console.ForegroundColor = ConsoleColor.Yellow;
            Console.WriteLine("Список книг:");
            foreach (var item in library.Books)
            {
                Console.WriteLine(item);
            }

            Console.ForegroundColor = ConsoleColor.Green;
            Console.WriteLine("~~~~~");
            Console.WriteLine("Книги старше 1990 года:");
            foreach (var item in library.GetBooksElderThan(1990))
            {
                Console.WriteLine(item);
            }

            Console.ForegroundColor = ConsoleColor.Cyan;
            Console.WriteLine("~~~~~");
            Console.WriteLine("Книги взятые на чтение:");
            foreach (var item in library.GetTakenForReading())
            {
                Console.WriteLine(item);
            }
        }
    }
}

```

```

Console.ForegroundColor = ConsoleColor.White;

Console.WriteLine("~~~~~");

Console.WriteLine("Книги взятые на чтение. Информация о читателях:");
library.GetTakenForReadingWithInfo();


Console.ForegroundColor = ConsoleColor.DarkYellow;

Console.WriteLine("~~~~~");

Console.WriteLine("Задержанные книги:");
foreach (var item in library.GetDelayedBooks())
{
    Console.WriteLine(item);
}


Console.ForegroundColor = ConsoleColor.White;
}
}

class Library
{
    private List<Book> books = new List<Book>();

    public IReadOnlyCollection<Book> Books { get => books; }

    public void AddBook(Book book) => books.Add(book);

    public Library(string filePath) => ReadInfoFromFile(filePath);

    public IEnumerable<Book> GetBooksElderThan(int publishYear) => books.Where(x => x.PublishYear >
publishYear);

    public IEnumerable<Book> GetTakenForReading() => books.Where(x =>
x.BookDeliveryDeadline.HasValue);

    public IEnumerable<Book> GetTakenForReadingWithInfo()
    {
        IEnumerable<Book> result = books.Where(x => x.BookDeliveryDeadline.HasValue);

        foreach (Book book in result)
        {
            Console.WriteLine($"Rader name: {book.ReaderFullName}");
        }

        return result;
    }
}

```

```

    }

    public IEnumerable<Book> GetDelayedBooks() => books.Where(x =>
x.BookDeliveryDeadline.HasValue).Where(x => DateTime.Now > x.BookDeliveryDeadline);

```

```

public void ReadInfoFromFile(string path)
{
    XmlSerializer formatter = new XmlSerializer(typeof(Book[]));
    using (FileStream fs = new FileStream(path, FileMode.Open))
    {
        books = ((Book[])formatter.Deserialize(fs)).ToList();
    }
}

```

```

public void WriteInfoInFile(string path, bool overwrite = true)
{
    XmlSerializer formatter = new XmlSerializer(typeof(Book[]));
    using (FileStream fs = new FileStream(path, FileMode.OpenOrCreate))
    {
        formatter.Serialize(fs, books.ToArray());
    }
}

```

[Serializable]

```

public class Book
{
    public int UDKNumber { get; set; }
    public string FullAuthorName { get; set; }
    public string Name { get; set; }
    public int PublishYear { get; set; }
    public int InstancesCount { get; set; }
    public int PagesCount { get; set; }
    public int VolumesCount { get; set; }
    public string ReaderFullName { get; set; }
    public DateTime? BookDeliveryDeadline { get; set; }
    public Book()

```



```
{
}

public override string ToString()
{
    StringBuilder result = new StringBuilder();
    result.Append($"UDKNumber: {UDKNumber},\n");
    result.Append($"Author name: {FullAuthorName},\n");
    result.Append($"Name: {Name},\n");
    result.Append($"Publish year: {PublishYear},\n");
    result.Append($"Instances: {InstancesCount},\n");
    result.Append($"Pages count: {PagesCount},\n");
    result.Append($"Volumes: {VolumesCount},\n");
    result.Append($"Reader: {ReaderFullName},\n");
    result.Append($"Book delivery deadline: {BookDeliveryDeadline};\n");
    return result.ToString();
}
}
```

Содержимое файла, из которого производилось чтение:

```

1 <?xml version="1.0" encoding="utf-8"?>
2 <ArrayOfBook xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
3   <Book>
4     <UDKNumber>1244</UDKNumber>
5     <FullAuthorName>Jeffrey Richter</FullAuthorName>
6     <Name>CLR via C#</Name>
7     <PublishYear>2012</PublishYear>
8     <InstancesCount>45</InstancesCount>
9     <PagesCount>896</PagesCount>
10    <VolumesCount>1</VolumesCount>
11    <BookDeliveryDeadline xsi:nil="true" />
12  </Book>
13  <Book>
14    <UDKNumber>12543</UDKNumber>
15    <FullAuthorName>Arthur Conan Doyle</FullAuthorName>
16    <Name>The Adventures Of Sherlock Holmes</Name>
17    <PublishYear>1886</PublishYear>
18    <InstancesCount>12</InstancesCount>
19    <PagesCount>2300</PagesCount>
20    <VolumesCount>4</VolumesCount>
21    <ReaderFullName>Petrov Ivan Petrovich</ReaderFullName>
22    <BookDeliveryDeadline>2020-09-14T12:11:23.2792947+03:00</BookDeliveryDeadline>
23  </Book>
24  <Book>
25    <UDKNumber>13244</UDKNumber>
26    <FullAuthorName>Andrzej Sapkowski</FullAuthorName>
27    <Name>Witcher</Name>
28    <PublishYear>1986</PublishYear>
29    <InstancesCount>4</InstancesCount>
30    <PagesCount>2824</PagesCount>
31    <VolumesCount>4</VolumesCount>
32    <ReaderFullName>Ivanov Ivan Ivanovich</ReaderFullName>
33    <BookDeliveryDeadline>2021-09-14T12:11:23.2792947+03:00</BookDeliveryDeadline>
34  </Book>
35 </ArrayOfBook>

```

Результаты работы программы:

Список книг:

UDKNumber: 1244,
Author name: Jeffrey Richter,
Name: Clr via C#,
Publish year: 2012,
Instances: 45,
Pages count: 896,
Volumes: 1,
Reader: ,
Book delivery deadline: ;

UDKNumber: 12543,
Author name: Arthur Conan Doyle,
Name: The Adventures Of Sherlock Holmes,
Publish year: 1886,
Instances: 12,
Pages count: 2300,
Volumes: 4,
Reader: Petrov Ivan Petrovich,
Book delivery deadline: 14.09.2020 12:11:23;

UDKNumber: 13244,
Author name: Andrzej Sapkowski,
Name: Witcher,
Publish year: 1986,
Instances: 4,
Pages count: 2824,
Volumes: 4,
Reader: Ivanov Ivan Ivanovich,
Book delivery deadline: 14.09.2021 12:11:23;

~~~~~  
Книги старше 1990 года:

UDKNumber: 1244,  
Author name: Jeffrey Richter,  
Name: Clr via C#,  
Publish year: 2012,  
Instances: 45,  
Pages count: 896,  
Volumes: 1,  
Reader: ,  
Book delivery deadline: ;

~~~~~  
Книги взятые на чтение:

UDKNumber: 12543,

Author name: Arthur Conan Doyle,

Name: The Adventures Of Sherlock Holmes,

Publish year: 1886,

Instances: 12,

Pages count: 2300,

Volumes: 4,

Reader: Petrov Ivan Petrovich,

Book delivery deadline: 14.09.2020 12:11:23;

UDKNumber: 13244,

Author name: Andrzej Sapkowski,

Name: Witcher,

Publish year: 1986,

Instances: 4,

Pages count: 2824,

Volumes: 4,

Reader: Ivanov Ivan Ivanovich,

Book delivery deadline: 14.09.2021 12:11:23;

~~~~~  
Книги взятые на чтение. Информация о читателях:

Rader name: Petrov Ivan Petrovich

Rader name: Ivanov Ivan Ivanovich  
~~~~~

Задержанные книги:

UDKNumber: 12543,

Author name: Arthur Conan Doyle,

Name: The Adventures Of Sherlock Holmes,

Publish year: 1886,

Instances: 12,

Pages count: 2300,

Volumes: 4,

Reader: Petrov Ivan Petrovich,

Book delivery deadline: 14.09.2020 12:11:23;