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

Факультет Инфокоммуникационных Технологий

**Лабораторная работа №7**

Выполнил:

Зенин Д.Д.

Проверил Иванов С. Е.

Санкт-Петербург,

2024

Оглавление

[ВВЕДЕНИЕ 3](#_Toc180165781)

[1. Упражнение 1 4](#_Toc180165782)

[2. Упражнение 2 6](#_Toc180165786)

[3. Упражнение 3 10](#_Toc180165792)

[4. Упражнение 4 12](#_Toc180165793)

[5. Упражнение 5 15](#_Toc180165794)

[6. Упражнение 6 17](#_Toc180165795)

[7. Упражнение 7 19](#_Toc180165796)

[ЗАКЛЮЧЕНИЕ 22](#_Toc180165797)

# ВВЕДЕНИЕ

Цель лабораторной работы №7 состояла в изучении наследования как важного элемента объектно-ориентированного программирования и приобретении навыков реализации иерархии классов.

# Упражнение 1

# В этом упражнении мы будем использовать наследование для построения иерархии между классами, имеющих отношение типа «является».

# Напишем код класса Item, который хранит информацию об объекте:

# 

Рисунок 1 – Код класса Item

Теперь наследуем класс Book от класса Item и добавим слово new в переопределенные в классе Book методы

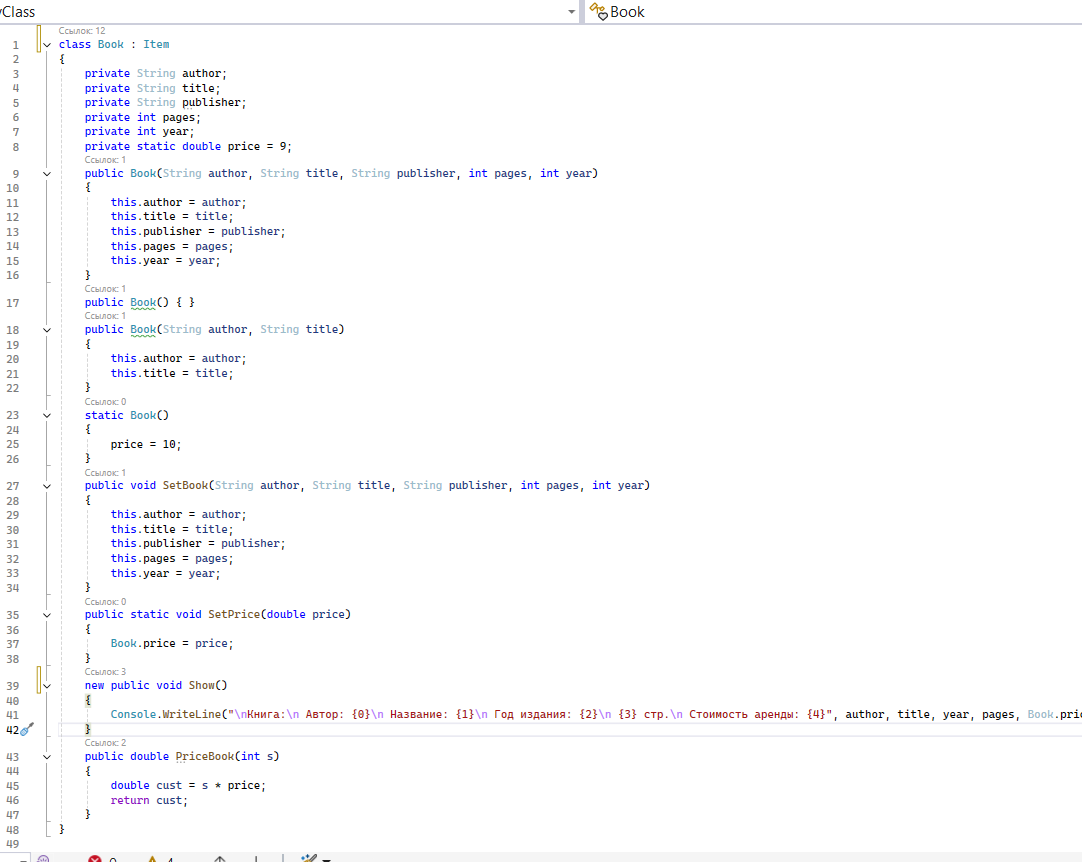


Рисунок 2 – Измененный код класса Book

Теперь в главном методе добавим объект класса Item и выведем его состояние

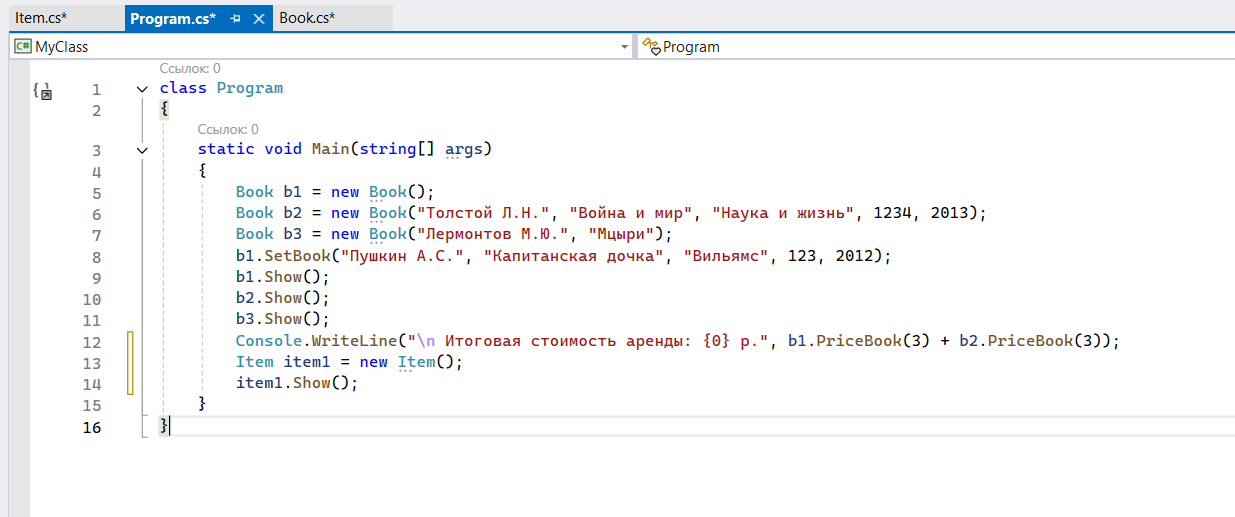


Рисунок 3 – Код главного метода

Запустим приложение

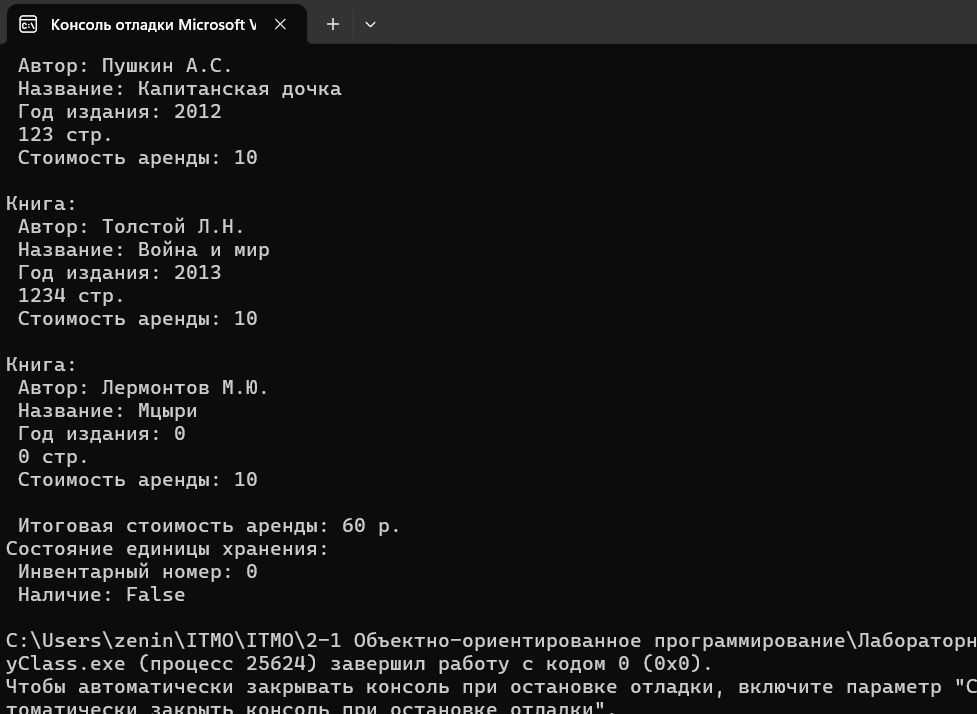


Рисунок 3 – Код главного метода

В графе «наличие» выводит False, что является ошибкой. В следующем упражнении исправим ее.

# Упражнение 2

# В этом упражнении определим конструкторы в базовом и производном классах и реализуем их выполнение в обоих классах с помощью ключевого слова base

# Добавим в класс Item новый конструктор

# 

Рисунок 4 – Конструктор класса Item

Далее в классе Book создадим конструктор, принимающий как свои параметры, так и параметры класса-родителя

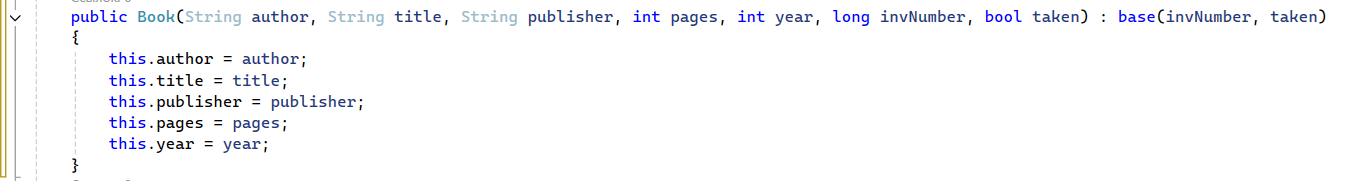


Рисунок 5 – Конструктор класса Book

В главном методе изменим создание переменной b2, добавив в конструктор два новых параметра и запустим два метода этого объекта

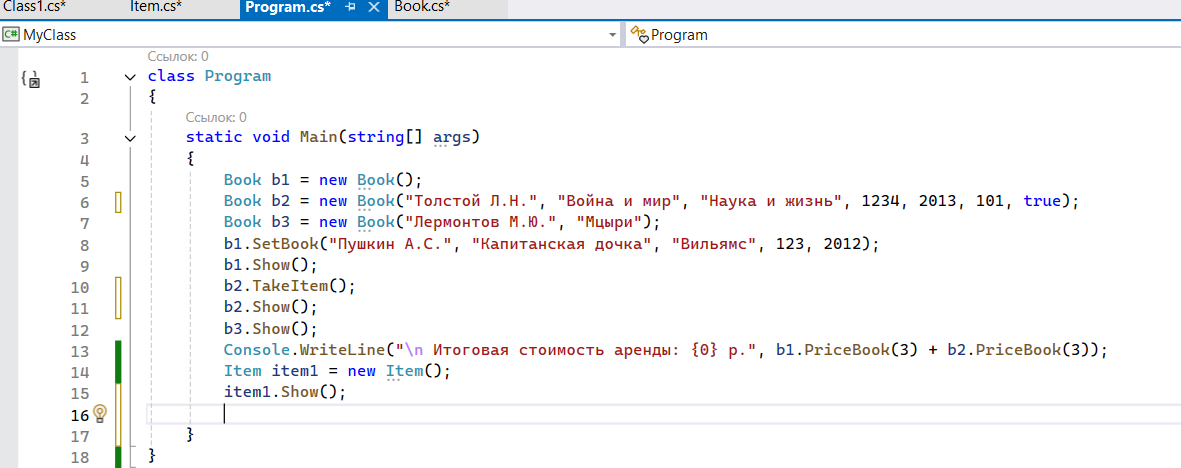


Рисунок 6 – Код главного метода

# Запустим программу

# 

Рисунок 7 – Результат работы программы

Все работает правильно.

Далее создадим класс Magazine, которые сделаем наследником класса Item

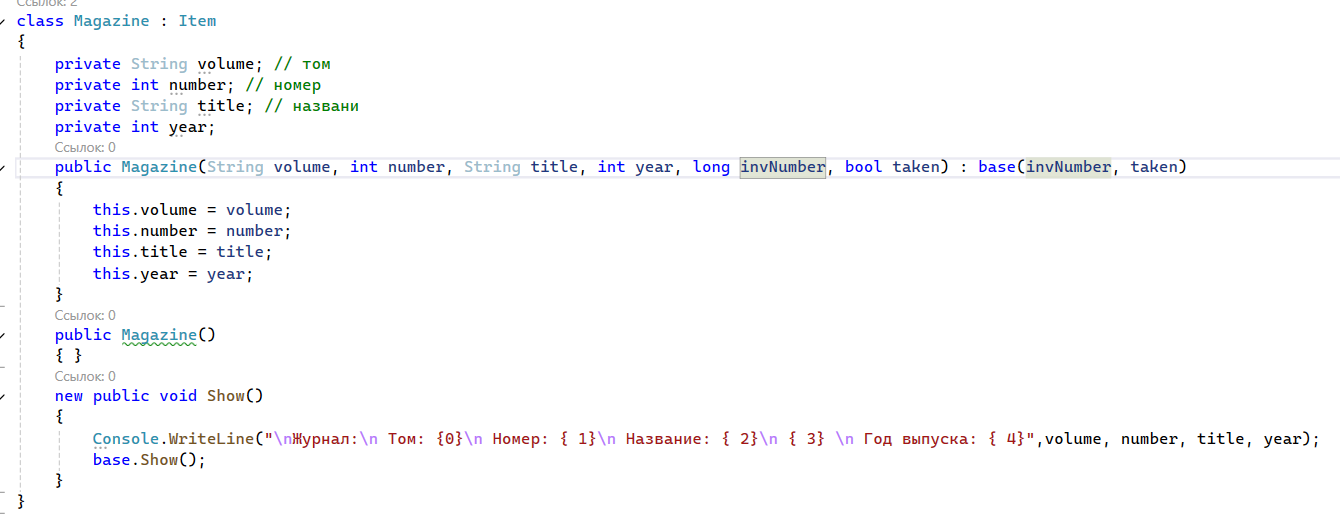
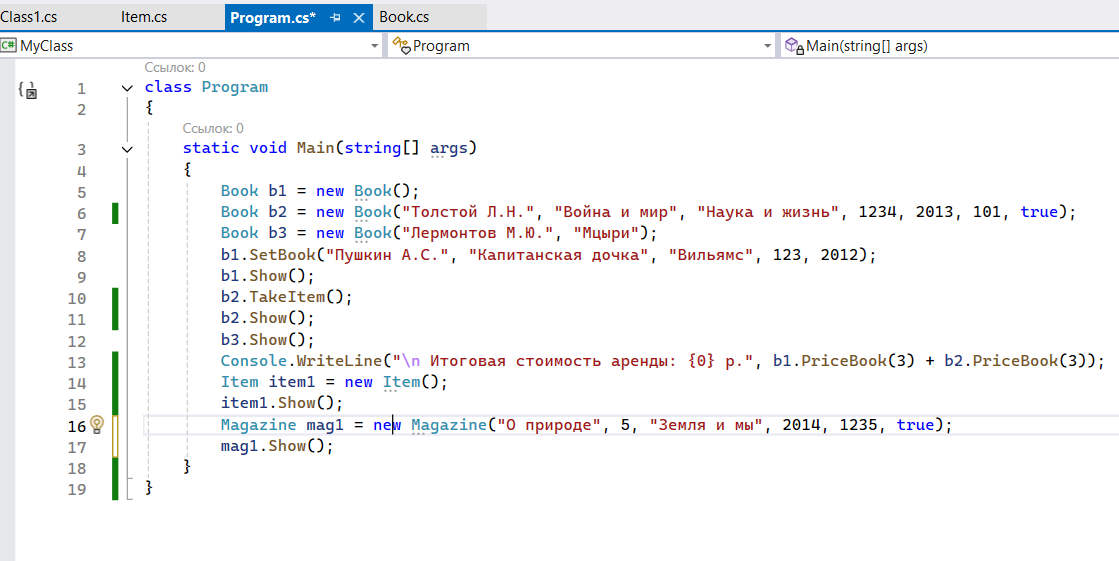


Рисунок 8 – Код класса Magazine

Добавим создание экземпляра класса Magazine в главном методе и запустим метод Show()

Рисунок 9 – Код главного метода

Запустим программу

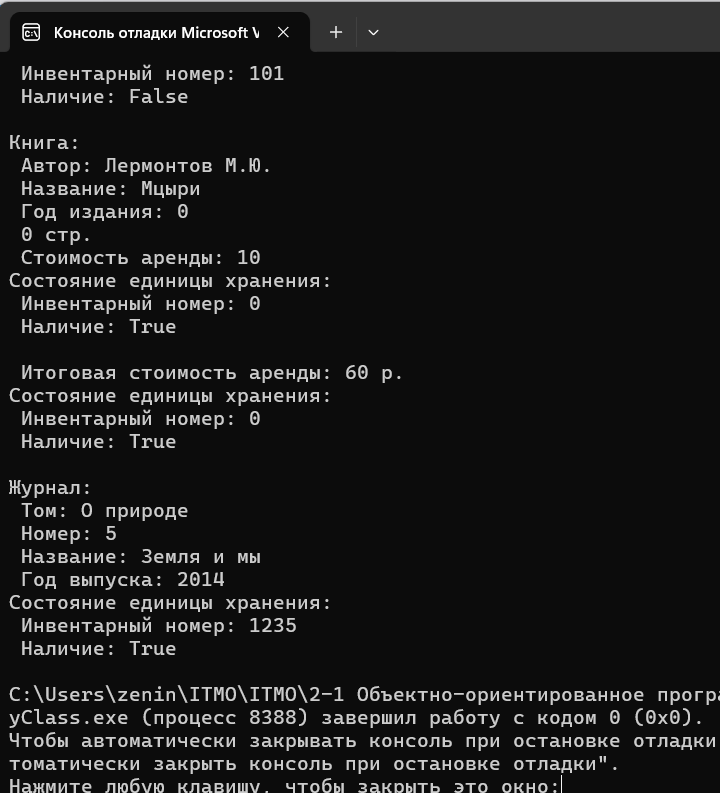


Рисунок 10 – Вывод программы

# Упражнение 3

В этом упражнении реализуем механизм полиморфизма с помощью виртуальных методов и их переопределения в производных классах.

В классе Item добавим к методам приписку virtual

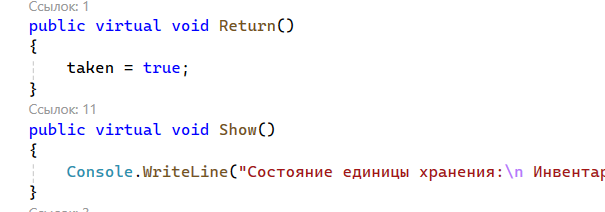
****

Рисунок 11 – Методы класса Item

Далее в классе Book и в классе magazine добавим в методы приписку override, чтобы методы переопределялись явно

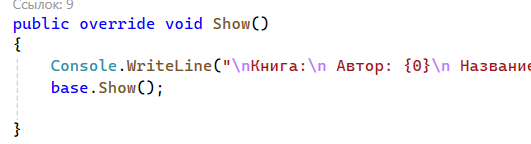


Рисунок 12 – Пример изменения метода

В главном методе создадим объект Item и присвоим ему объект дочернего класса. Вызовем разные методы и посмотрим, что будет

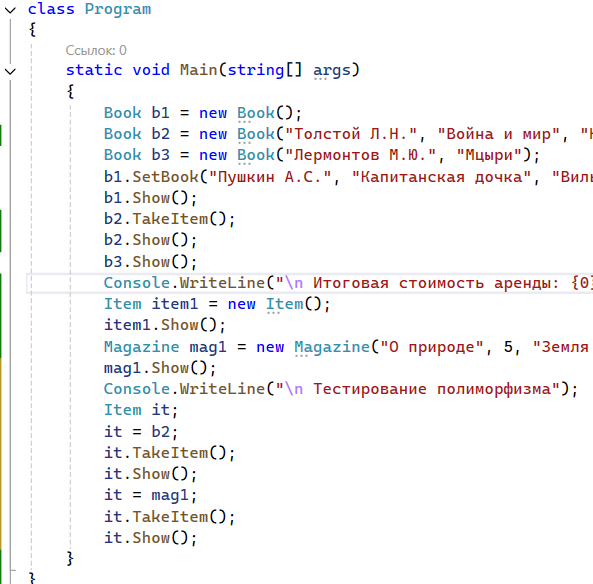


Рисунок 12 – Пример изменения метода

Запустим программу

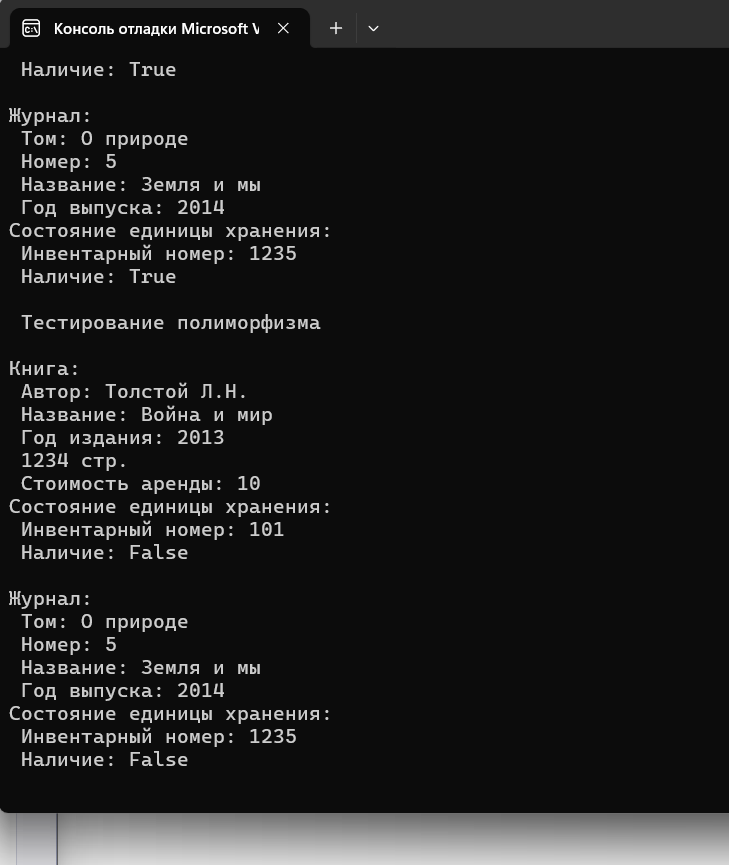


Рисунок 13 – Результат работы программы

Упражнение выполнено.

# Упражнение 4

В упражнении создадим базовый класс и сделаем один из его методов абстрактным.

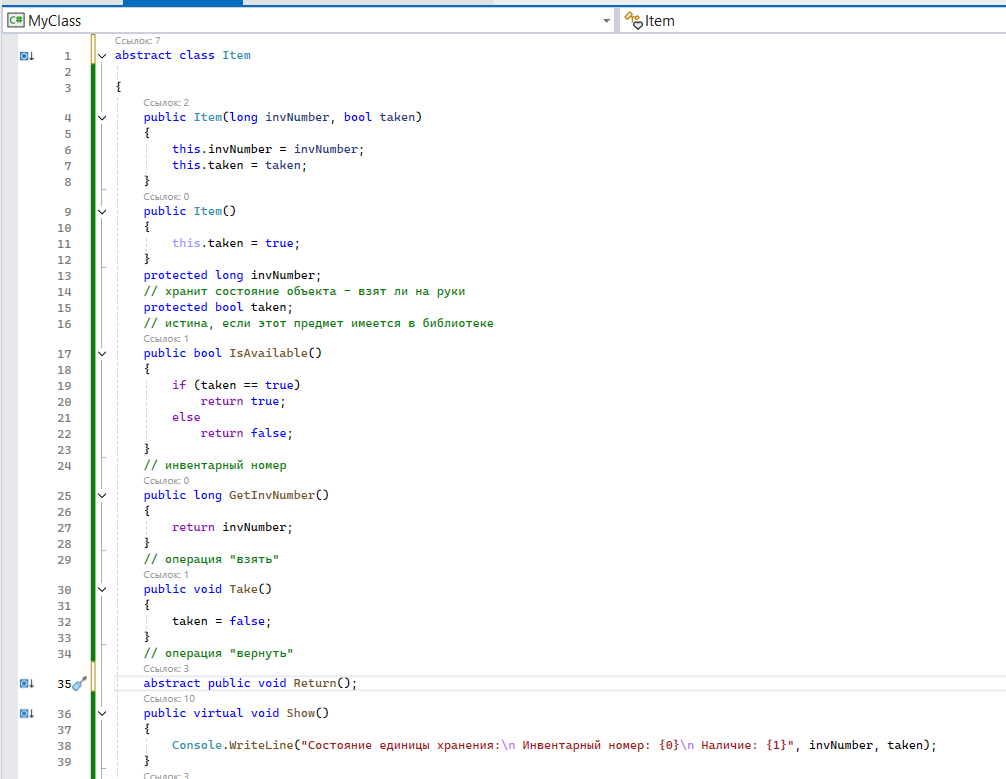


Рисунок 14 – Абстрактный класс Item

В главном методе вызовем абстрактный метод

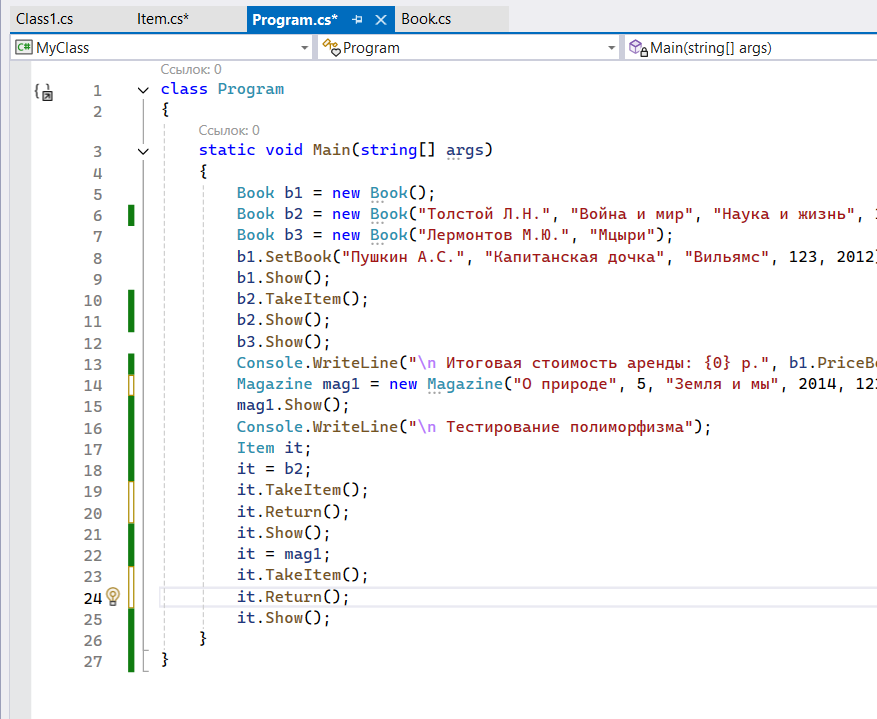


Рисунок 15 – Главный метод программы

Запустим программу

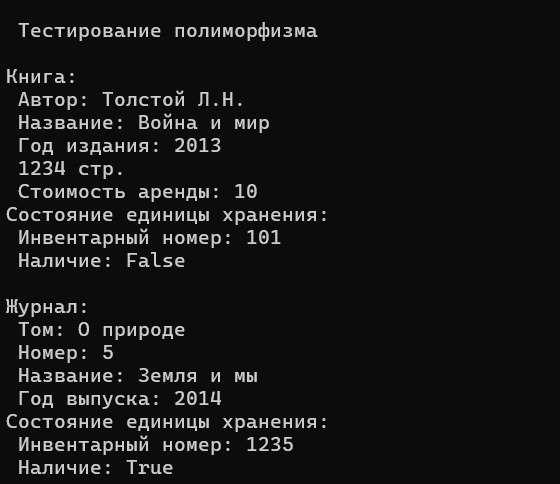


Рисунок 16 – Вывод программы

Упражнение сделано.

# Упражнение 5

В этом упражнении реализуем отношение «имеет» (агрегация).

Создадим в новом проекте класс Point

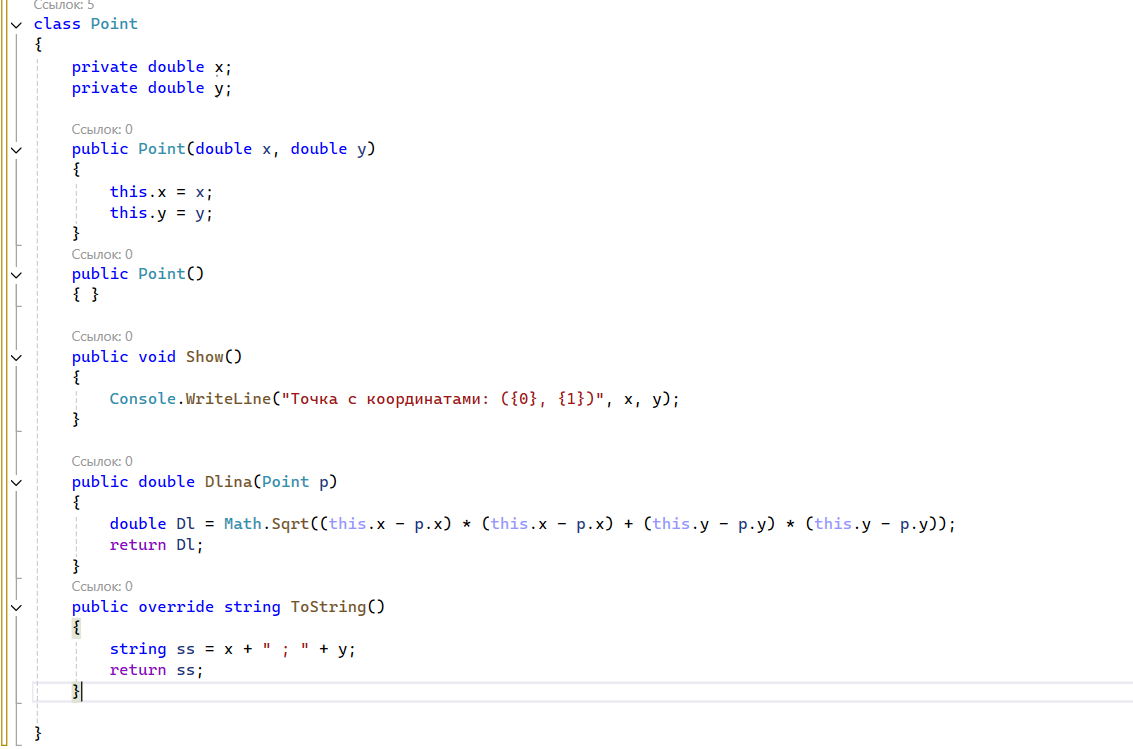
****

Рисунок 17 – Класс Point

Далее напишем агрегирующий от Point класс Line

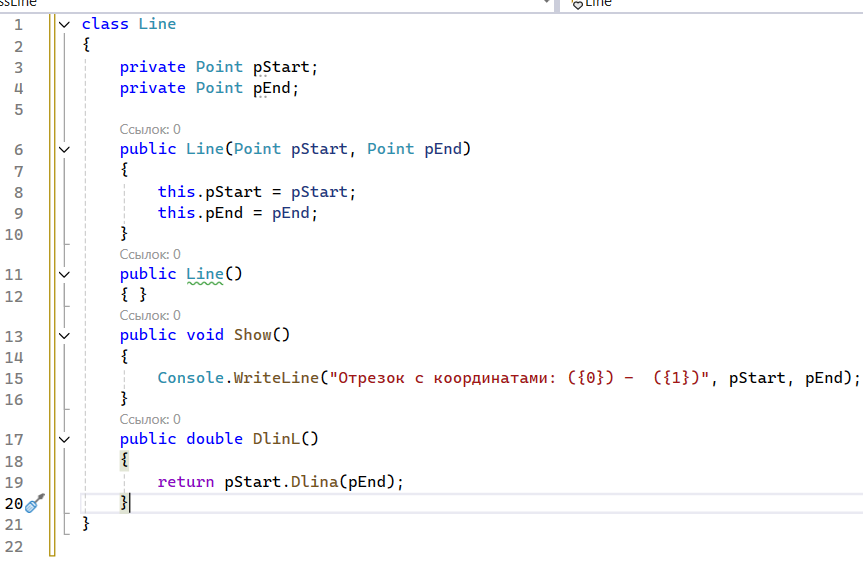


Рисунок 18 – Класс Line

Далее в главном методе создадим объекты классов Line и Point и выведем информацию о них

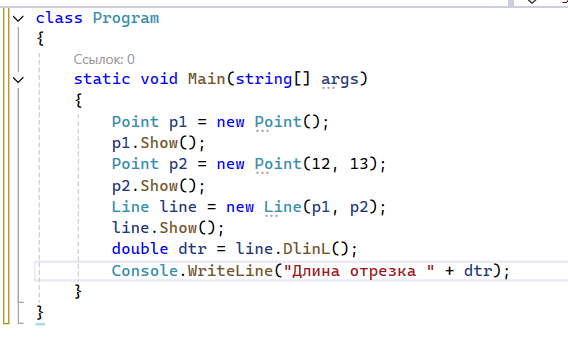


Рисунок 19 – Код программы

Запустим программу

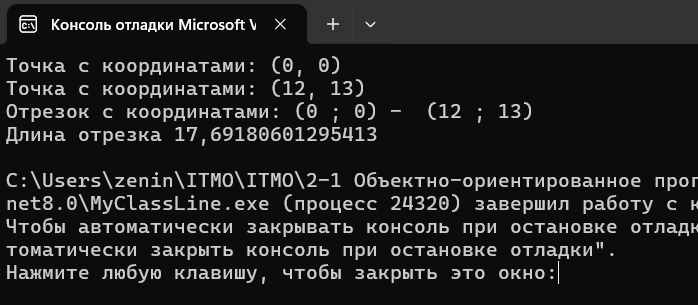


Рисунок 20 – Результат работы программы

# Упражнение 6

В этом упражнении создадим класс, моделирующий игральную кость и класс игрока, который связан с классом игры отношением ассоциации

Создадим класс игральной кости

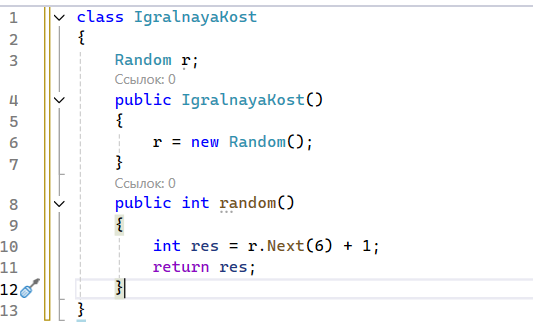
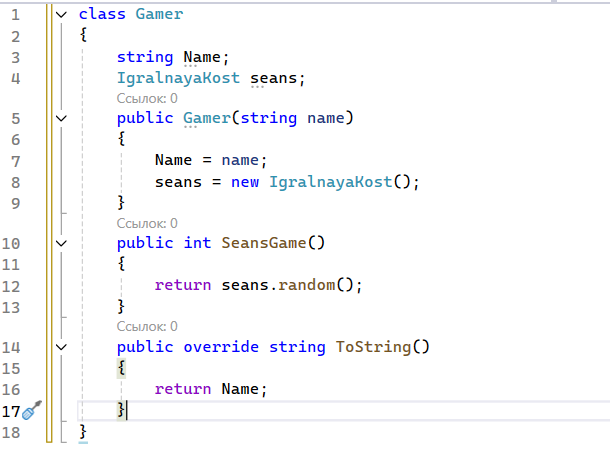


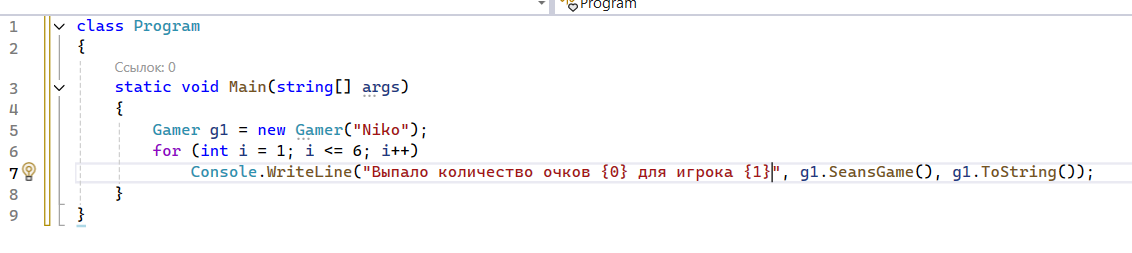
Рисунок 21 – Класс IgralnayaKost

Далее создадим класс игрока



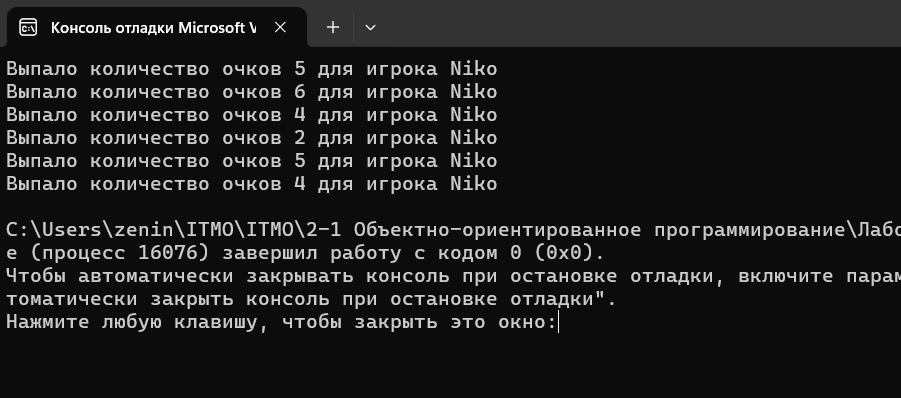
*Рисунок 22 – Класс Gamer*

В главном классе создадим игрока и бросим кость



*Рисунок 23 – Код программы*

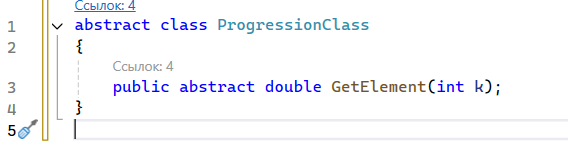
Запустим программу



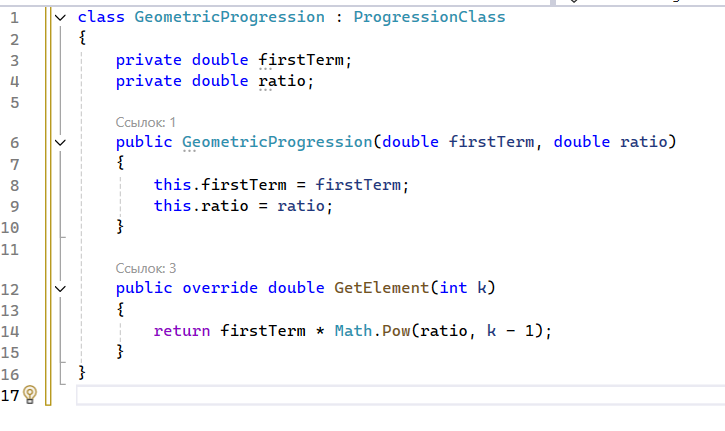
*Рисунок 24 – Результат работы программы*

# Упражнение 7

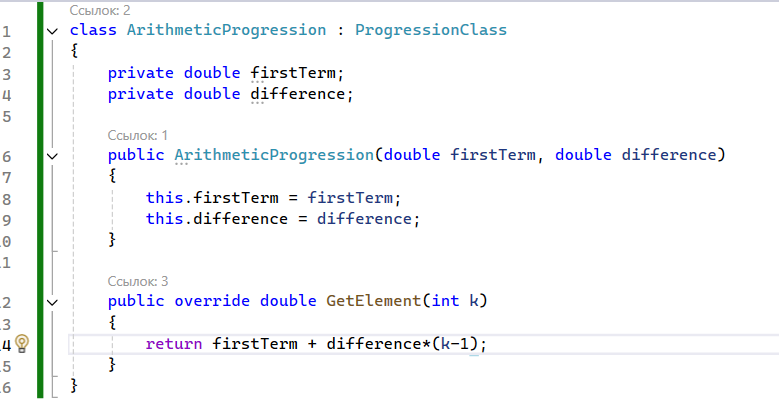
В этом упражнении реализуем абстрактный класс прогрессии, от которого будут наследоваться класс арифметической и геометрической прогрессий.



*Рисунок 25 – Абстрактный класс прогрессии*

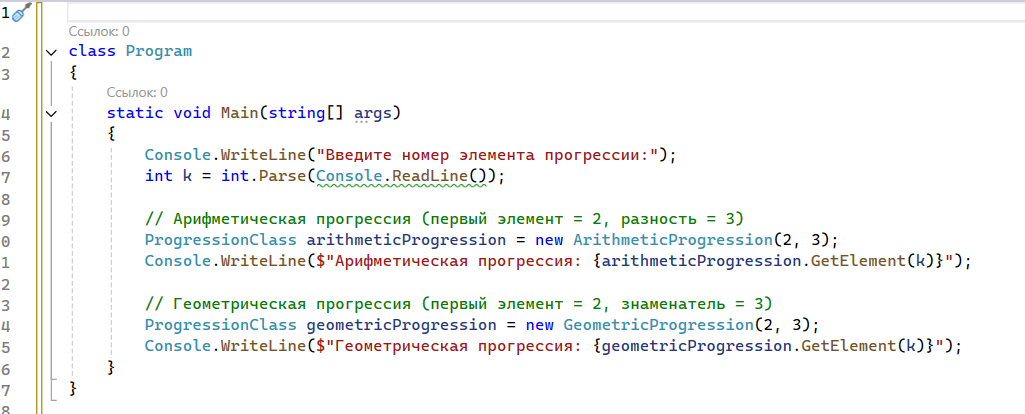
**

*Рисунок 26 – Класс геометрической прогрессии*

**

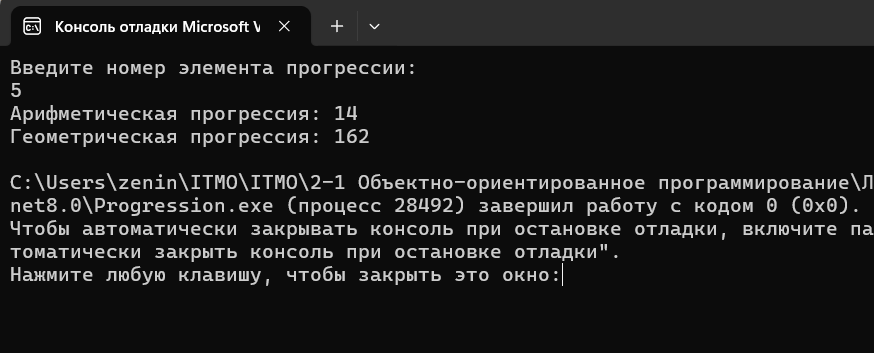
*Рисунок 27 – Класс арифметической прогрессии*

В главном методе создадим объекты класса и запустим их методы



*Рисунок 28 – Главный класс программы*

Запустим программу



*Рисунок 29 – Результат работы программы*

# ЗАКЛЮЧЕНИЕ

В ходе выполнения лабораторной работы №7 были получены навыки работы с языком программирования С#