

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

Факультет инфокоммуникационных технологий

Лабораторная работа №9
Применение делегатов и событий

Выполнил
Стафеев И.А.

Группа
К3221

Проверил
Иванов С.Е.

Санкт-Петербург,
2024

СОДЕРЖАНИЕ

	Стр.
ВВЕДЕНИЕ	3
1 Упражнение 1.....	4
2 Упражнение 2.....	6
3 Упражнение 3.....	9
ЗАКЛЮЧЕНИЕ	12

ВВЕДЕНИЕ

Цель работы: изучение делегатов и событий в языке C# и приобретение навыков работы с ними.

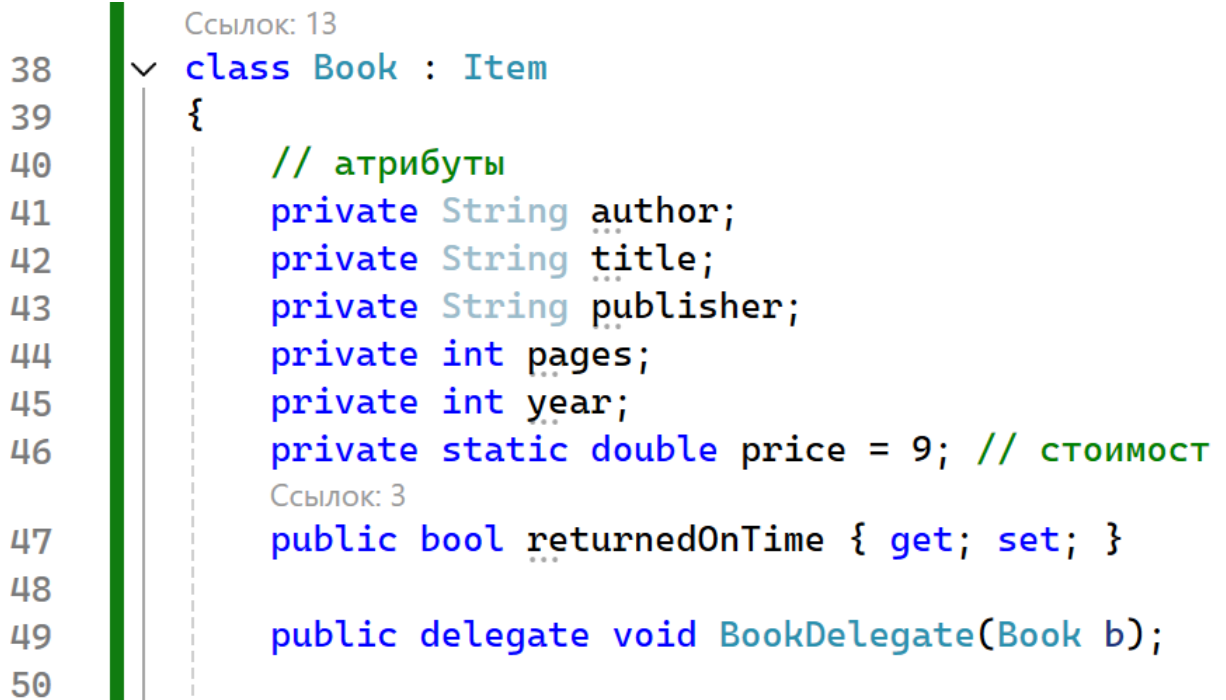
Для достижения цели необходимо выполнить следующие упражнения:

1. Использование делегата при вызове метода
2. Работа с событиями
3. Реализация события

1 Упражнение 1

Задача: добавить реализацию вывода информации о книгах, возвращенных в срок, посредством использования делегатов.

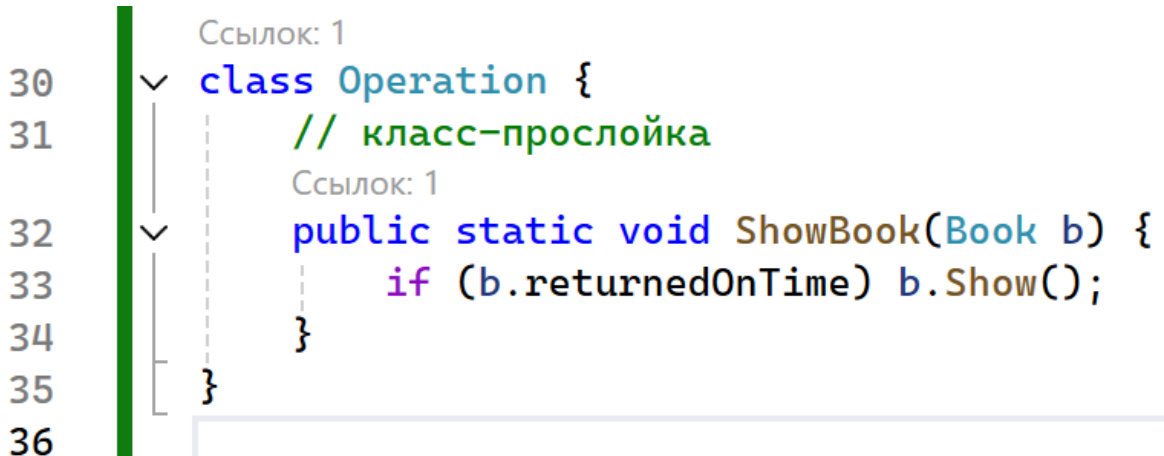
В классе Book поле returnedOnTime сделано публичным свойством, а также добавлен делегат BookDelegate (1).



```
38  class Book : Item
39  {
40      // атрибуты
41      private String author;
42      private String title;
43      private String publisher;
44      private int pages;
45      private int year;
46      private static double price = 9; // стоимость
47      public bool returnedOnTime { get; set; }
48
49      public delegate void BookDelegate(Book b);
50
```

Рисунок 1 — Делегат BookDelegate

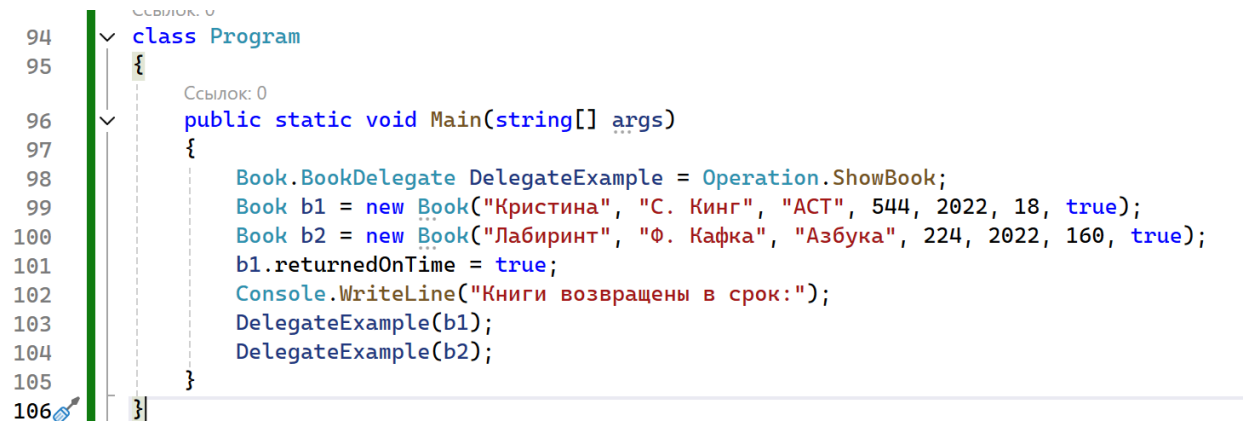
Метод, который будет вызываться делегатом, создан в отдельном классе Operation (2). Метод имеет ту же сигнатуру, что и делегат в классе Book.



```
30  class Operation {
31      // класс-прослойка
32      public static void ShowBook(Book b) {
33          if (b.returnedOnTime) b.Show();
34      }
35  }
36
```

Рисунок 2 — Класс Operation

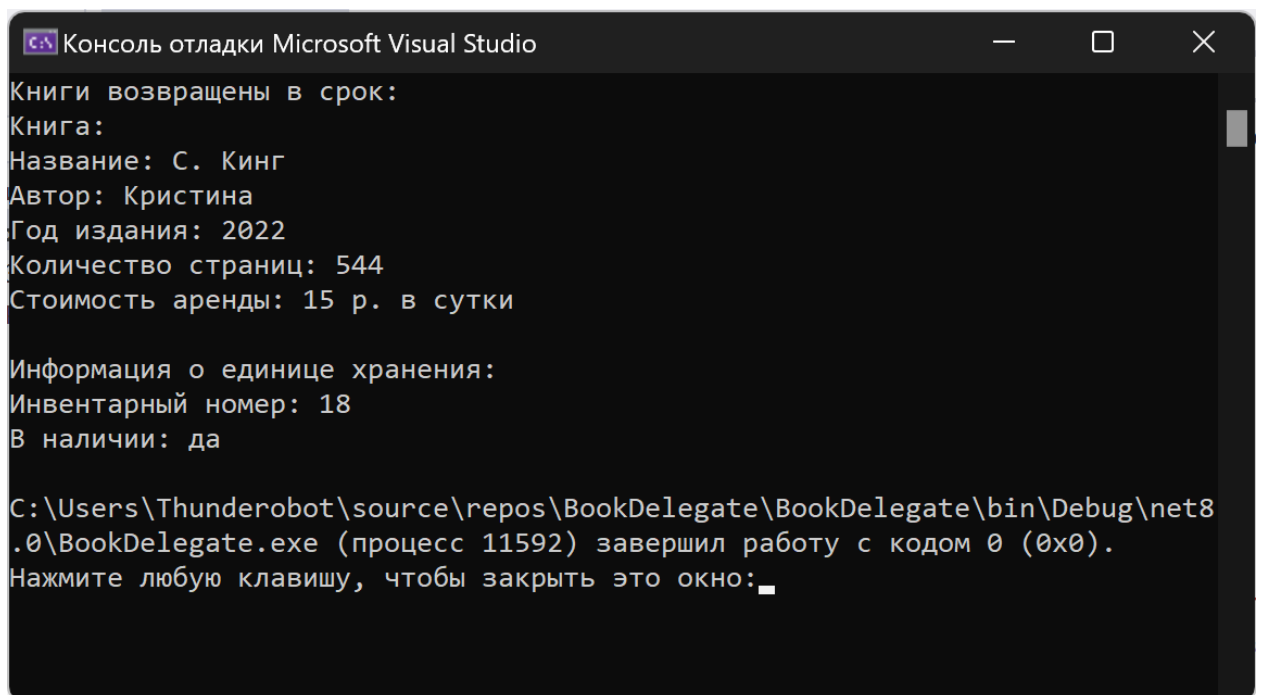
Метод Main представлен на рисунке 3. В нем создается экземпляр делегата, которому присваивается метод из созданного ранее класса Operation. Для теста созданы два экземпляра класса Book, один из которых считается возвращенным в срок. После для обеих книг вызывается делегат.



```
94 class Program
95 {
96     public static void Main(string[] args)
97     {
98         Book.BookDelegate DelegateExample = Operation.ShowBook;
99         Book b1 = new Book("Кристина", "С. Кинг", "АСТ", 544, 2022, 18, true);
100        Book b2 = new Book("Лабиринт", "Ф. Кафка", "Азбука", 224, 2022, 160, true);
101        b1.returnedOnTime = true;
102        Console.WriteLine("Книги возвращены в срок:");
103        DelegateExample(b1);
104        DelegateExample(b2);
105    }
106 }
```

Рисунок 3 — Метод Main

Результат выполнения программы показан на рисунке 4. Видно, что выведена информация только о первой книге, которая была сдана в срок.



```
Консоль отладки Microsoft Visual Studio
Книги возвращены в срок:
Книга:
Название: С. Кинг
Автор: Кристина
Год издания: 2022
Количество страниц: 544
Стоимость аренды: 15 р. в сутки

Информация о единице хранения:
Инвентарный номер: 18
В наличии: да

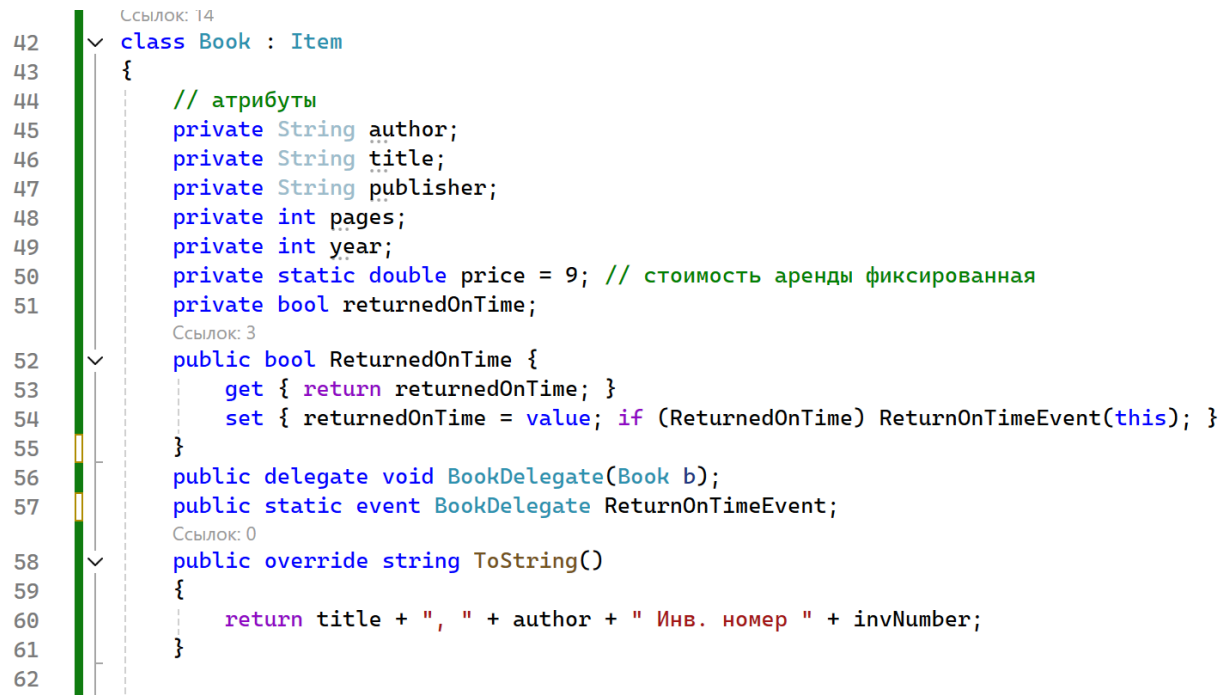
C:\Users\Thunderobot\source\repos\BookDelegate\BookDelegate\bin\Debug\net8
.0\BookDelegate.exe (процесс 11592) завершил работу с кодом 0 (0x0).
Нажмите любую клавишу, чтобы закрыть это окно: _
```

Рисунок 4 — Результат выполнения программы

2 Упражнение 2

Задача: реализовать вывод сообщений о возвращенных в срок книгах посредством событий.

В классе `Book` было создано приватное поле **`returnedOnTime`** и свойство **`ReturnedOnTime`**. В сеттере свойства при установке значения `true` вызывается событие **`ReturnOnTimeEvent`**, связанное с созданным ранее делегатом (см. рисунок 5).



```
42  class Book : Item
43  {
44      // атрибуты
45      private String author;
46      private String title;
47      private String publisher;
48      private int pages;
49      private int year;
50      private static double price = 9; // стоимость аренды фиксированная
51      private bool returnedOnTime;
52
53      public bool ReturnedOnTime {
54          get { return returnedOnTime; }
55          set { returnedOnTime = value; if (ReturnedOnTime) ReturnOnTimeEvent(this); }
56      }
57      public delegate void BookDelegate(Book b);
58      public static event BookDelegate ReturnOnTimeEvent;
59
60      public override string ToString()
61      {
62          return title + ", " + author + " Инв. номер " + invNumber;
```

Рисунок 5 — Измененный класс `Book`

В класс `Operation` добавлен метод **`ProcessBook`**, сообщающий, что книга возвращена в срок (6).

```

--
30  class Operation {
31      // класс-прослойка
32      public static void ShowBook(Book b) {
33          if (b.ReturnedOnTime) b.Show();
34      }
35
36      public static void ProcessBook(Book b) {
37          Console.WriteLine("Книга {0} сдана в срок", b);
38      }
39  }
40

```

Рисунок 6 — Метод ProcessBook класса Operation

Код метода Main представлен на рисунке 7. В обработку созданного события добавлен метод ProcessBook класса Operation.

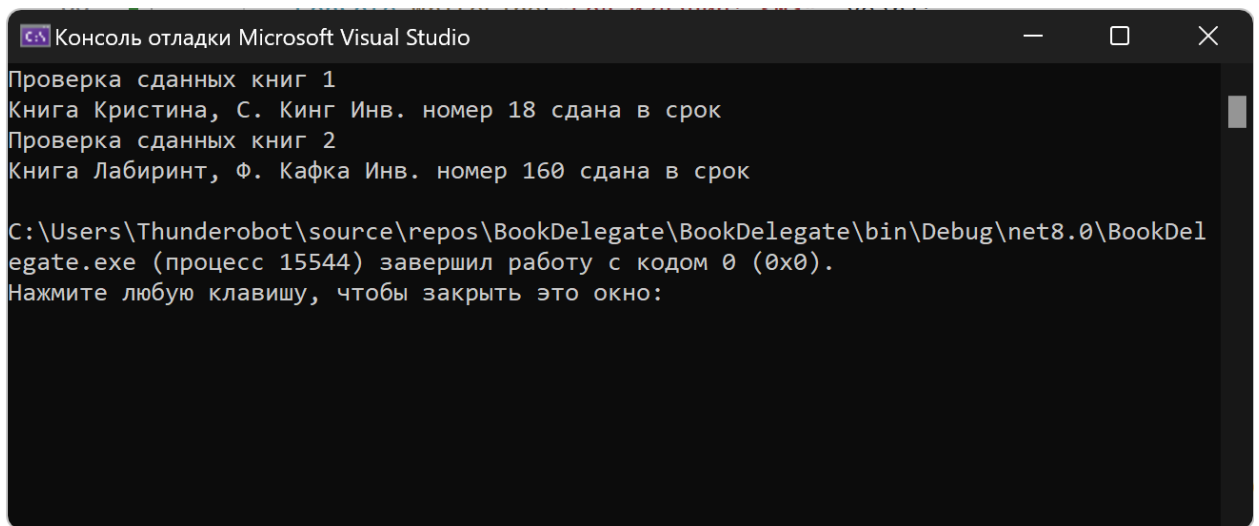
```

106  class Program
107  {
108      public static void Main(string[] args)
109      {
110          Book.ReturnOnTimeEvent += Operation.ProcessBook;
111          Book b1 = new Book("С. Кинг", "Кристина", "АСТ", 544, 2022, 18, true);
112          Book b2 = new Book("Ф. Кафка", "Лабиринт", "Азбука", 224, 2022, 160, true);
113          Console.WriteLine("Проверка сданных книг 1");
114          b1.ReturnedOnTime = true;
115          Console.WriteLine("Проверка сданных книг 2");
116          b2.ReturnedOnTime = true;
117      }
118  }

```

Рисунок 7 — Метод Main

Результат выполнения программы показан на рисунке 8. Видно, что после установки значения true у поля ReturnedOnTime у книг сразу вызывается обработка этого события, и выводится сообщение о сдаче в срок.



```
Консоль отладки Microsoft Visual Studio

Проверка сданных книг 1
Книга Кристина, С. Кинг Инв. номер 18 сдана в срок
Проверка сданных книг 2
Книга Лабиринт, Ф. Кафка Инв. номер 160 сдана в срок

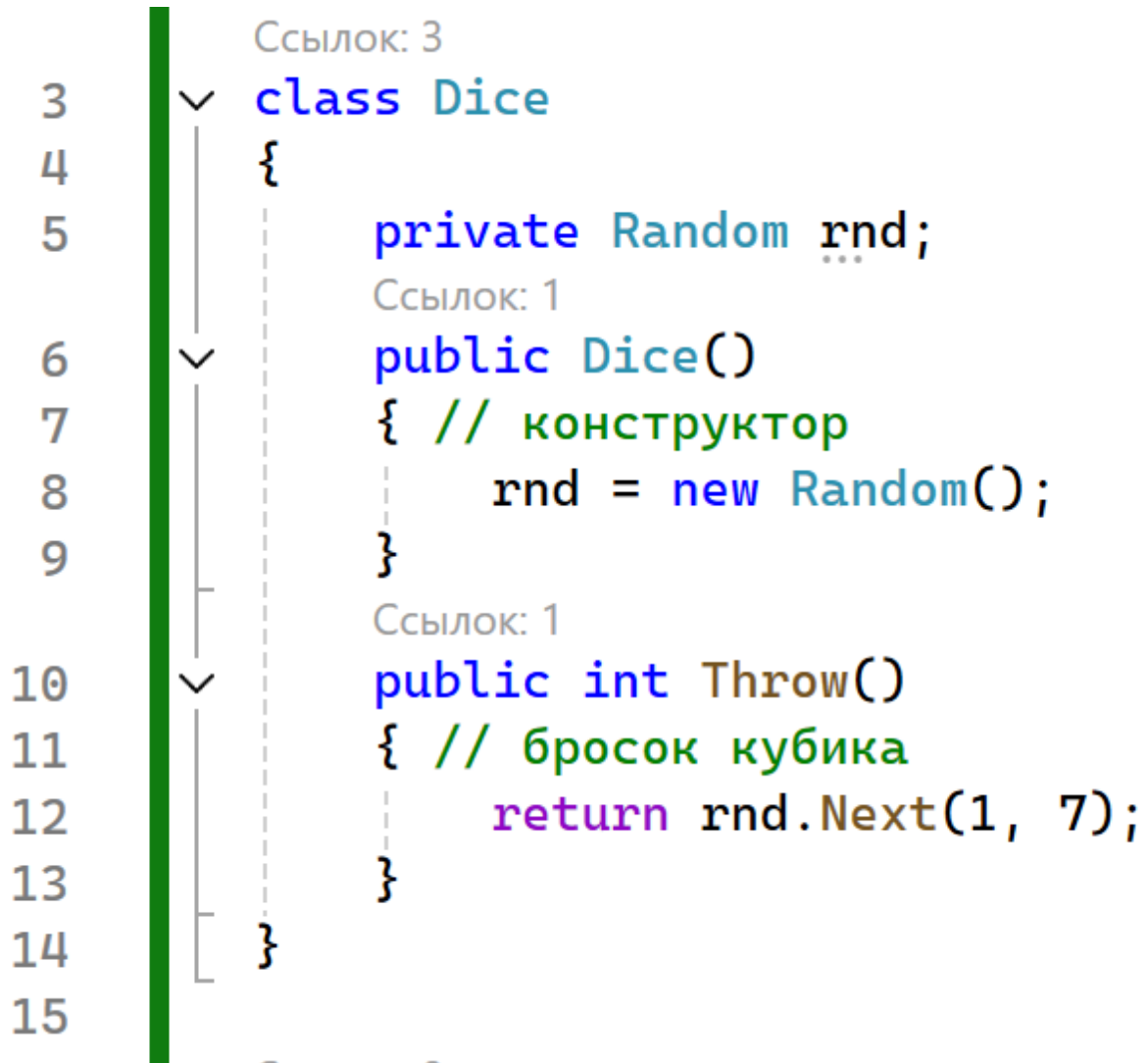
C:\Users\Thunderobot\source\repos\BookDelegate\BookDelegate\bin\Debug\net8.0\BookDelegate.exe (процесс 15544) завершил работу с кодом 0 (0x0).
Нажмите любую клавишу, чтобы закрыть это окно:
```

Рисунок 8 — Результат выполнения программы

3 Упражнение 3

Задача: реализовать событие "выпало максимальное количество очков" и его обработку для решения игровой кости из лабораторной работы №7.

Класс Dice остался без изменений (9).



```
3  class Dice
4  {
5      private Random rnd;
6      public Dice()
7      { // конструктор
8          rnd = new Random();
9      }
10     public int Throw()
11     { // бросок кубика
12         return rnd.Next(1, 7);
13     }
14 }
15
```

Ссылка: 3

Ссылка: 1

Ссылка: 1

Рисунок 9 — Класс Dice

В классе `Player` добавлен делегат `PlayerDelegate` и событие `MaxPointsEvent`. В методе `Play` при получении 6 очков вызывается созданное событие.

```

16  class Player
17  {
18      private string name;
19      private Dice dice;
20      public Player(string name)
21      {
22          this.name = name;
23          this.dice = new Dice();
24      }
25      public delegate void PlayerDelegate(Player player);
26      public static event PlayerDelegate MaxPointsEvent;
27      public int Play()
28      { // выполнить бросок кубика игроком
29          int res = dice.Throw();
30          if (res == 6) MaxPointsEvent(this);
31          return res;
32      }
33      public override string ToString()
34      {
35          return name;
36      }
37  }

```

Ссылки: 8, 2, 0

Рисунок 10 — Класс Player с делегатом и событием

В классе Program создан метод **ProcessPlayersThrow**, выводящий сообщения о том, что у игрока выпало максимальное количество очков. В Main к созданному событию добавляется этот метод.

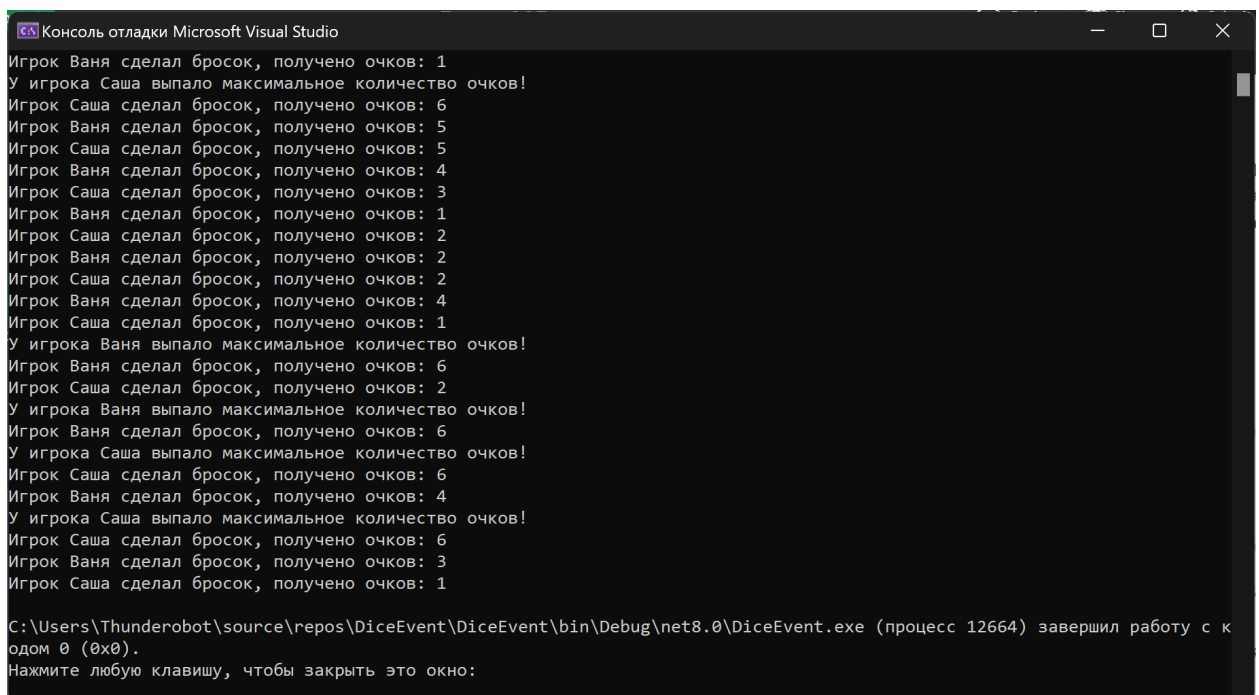
```

39  class Program
40  {
    Ссылка: 1
41      public static void ProcessPlayersThrow(Player p) {
42          Console.WriteLine("У игрока {0} выпало максимальное количество очков!", p);
43      }
    Ссылка: 0
44      public static void Main(string[] args)
45      {
46          Player.MaxPointsEvent += ProcessPlayersThrow;
47          Player p = new Player("Ваня");
48          Player p2 = new Player("Саша");
49          for (int i = 0; i < 10; i++)
50          {
51              Console.WriteLine("Игрок {0} сделал бросок, получено очков: {1}", p, p.Play());
52              Console.WriteLine("Игрок {0} сделал бросок, получено очков: {1}", p2, p2.Play());
53          }
54      }
55  }

```

Рисунок 11 — Класс Program

Пример выполнения программы показан на рисунке 12. Сообщение о максимальном количестве очков выводится раньше стандартного сообщения о полученных за бросок очках, что логично, поскольку обработка события происходит раньше, чем вызов `Console.WriteLine`.



```

Консоль отладки Microsoft Visual Studio
Игрок Ваня сделал бросок, получено очков: 1
У игрока Саша выпало максимальное количество очков!
Игрок Саша сделал бросок, получено очков: 6
Игрок Ваня сделал бросок, получено очков: 5
Игрок Саша сделал бросок, получено очков: 5
Игрок Ваня сделал бросок, получено очков: 4
Игрок Саша сделал бросок, получено очков: 3
Игрок Ваня сделал бросок, получено очков: 1
Игрок Саша сделал бросок, получено очков: 2
Игрок Ваня сделал бросок, получено очков: 2
Игрок Саша сделал бросок, получено очков: 2
Игрок Ваня сделал бросок, получено очков: 4
Игрок Саша сделал бросок, получено очков: 1
У игрока Ваня выпало максимальное количество очков!
Игрок Ваня сделал бросок, получено очков: 6
Игрок Саша сделал бросок, получено очков: 2
У игрока Ваня выпало максимальное количество очков!
Игрок Ваня сделал бросок, получено очков: 6
У игрока Саша выпало максимальное количество очков!
Игрок Саша сделал бросок, получено очков: 6
Игрок Ваня сделал бросок, получено очков: 4
У игрока Саша выпало максимальное количество очков!
Игрок Саша сделал бросок, получено очков: 6
Игрок Ваня сделал бросок, получено очков: 3
Игрок Саша сделал бросок, получено очков: 1
C:\Users\Thunderobot\source\repos\DiceEvent\DiceEvent\bin\Debug\net8.0\DiceEvent.exe (процесс 12664) завершил работу с кодом 0 (0x0).
Нажмите любую клавишу, чтобы закрыть это окно:

```

Рисунок 12 — Пример выполнения программы

ЗАКЛЮЧЕНИЕ

В ходе выполнения лабораторной работы были выполнены все требуемые упражнения. Цель работы достигнута. Получены знания о делегатах и событиях в языке C# и приобретены навыки работы с ними.