Модуль 3, практическое занятие 3

События

Задача 1

Описать событийный делегат Del.

В классе описать статическое событие Ev и определить три обработчика.

В основной программе связать с событием обработчики и активировать событие.

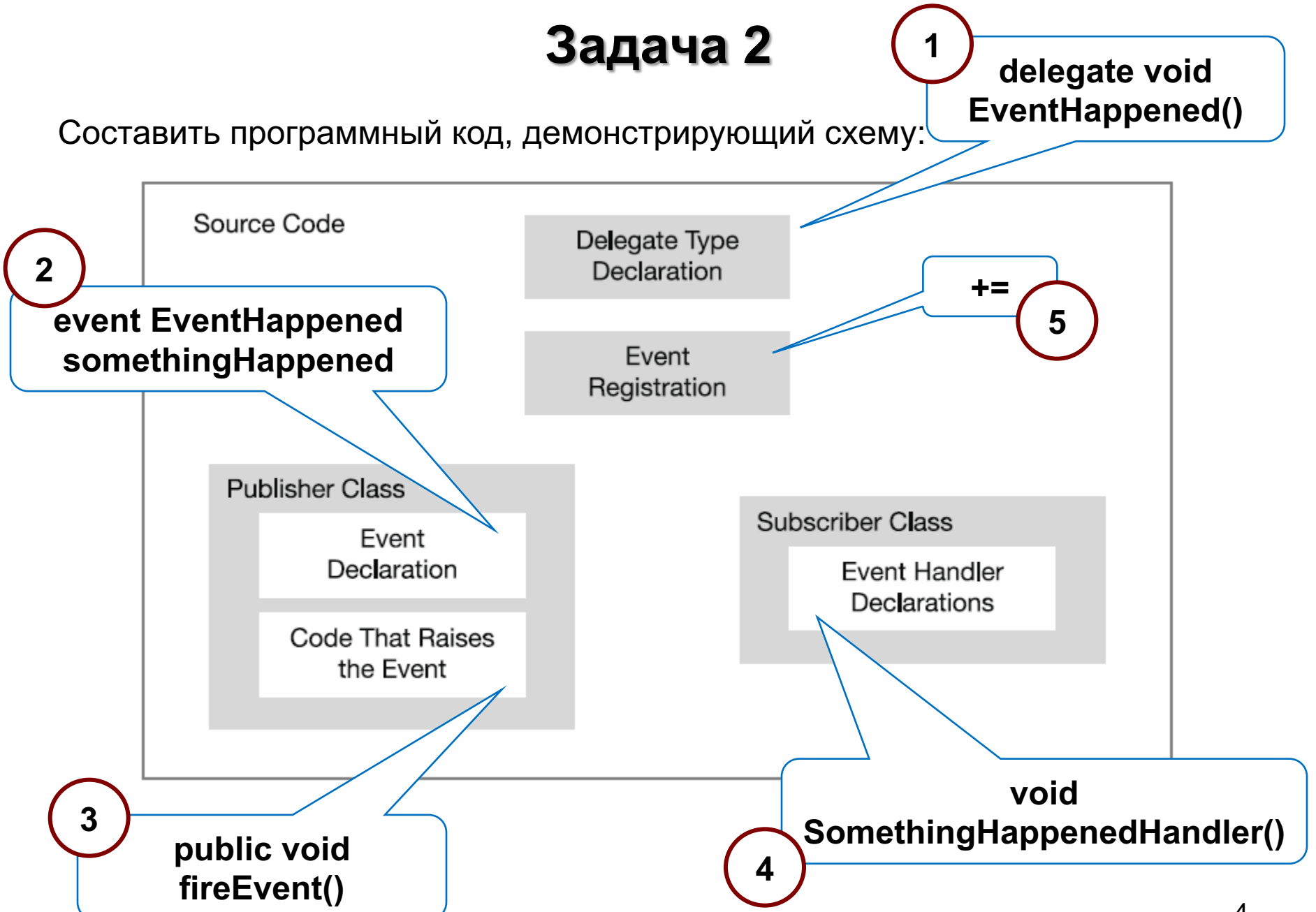
10.7.3 Static and instance events [https://msdn.microsoft.com/en-us/library/aa664457(v=vs.71).aspx**]**

Задача 1

```
using System;

delegate void Del(); // событийный делегат

class Program_0 {
    static event Del Ev; // событие

    // набор обработчиков
    static void f1() { Console.WriteLine("f1"); }
    static void f2() { Console.WriteLine("f2"); }
    static void f3() { Console.WriteLine("f3"); }

    static void Main() {
        Ev += f1; // добавление обработчика
        Ev += f3; // добавление обработчика
        Ev += f2; // добавление обработчика
        Ev();     // запуск события!
    }
}
```

Задача 2

Составить программный код, демонстрирующий схему:



Задача 2

```
public delegate void EventHappened(); // событийный делегат

public class Publisher { // класс-источник
    public event EventHappened somethingHappened; // событие

    public void fireEvent() {
        Console.WriteLine("Fire somethingHappened!!!");
        somethingHappened(); // запуск события!!!
    }
}

// класс-подписчик на somethingHappened
public class SomethingHappenedSubscriber {
    public void SomethingHappenedHandler() {
        // код обработки события
        Console.WriteLine("Subscriber has handled an event!");
    }
}
```

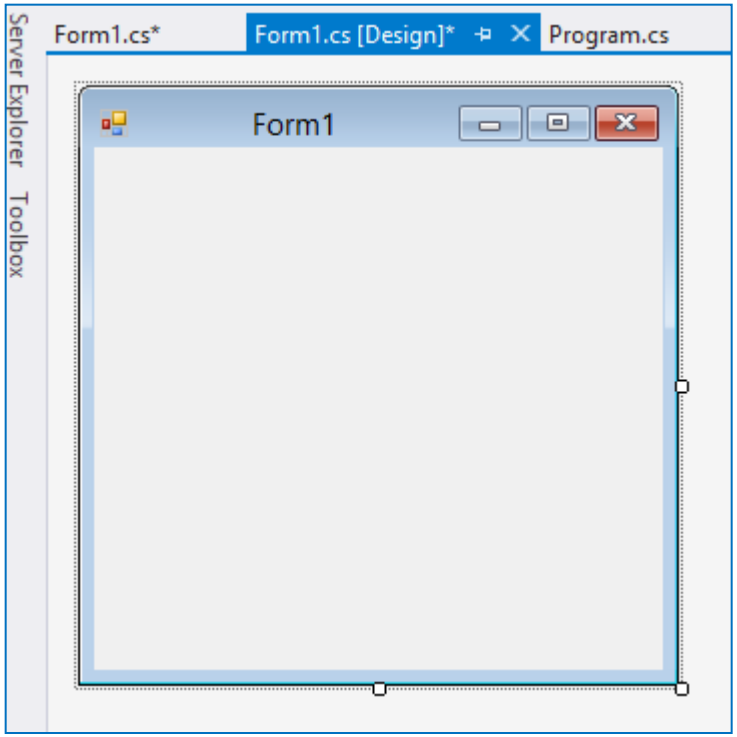
Задача 2

```
class Program {
    static void Main() {
        // объект класса-источника
        Publisher pub = new Publisher();
        // объект класса-подписчика
        SomethingHappenedSubscriber shs =
            new SomethingHappenedSubscriber();
        // добавляем подписчика к событию
        pub.somethingHappened += shs.SomethingHappenedHandler;
        // вызвали метод, запускающий событие
        pub.fireEvent();
    }
}
```

TODO: Добавить второй класс, в который поместить обработчик события somethingHappened

Задача 3

1) Создайте проект оконного приложения



Задача 3

1) Перейдите в код формы (**F7**)
2) Добавьте в описание класса статическое поле **counter** – счётчик входов мыши в форму
3) В конструкторе свяжите событие «Вход мыши» (**OnMouseEnter**) с обработчиком, передаваемым при помощи лямбда-выражения. В лямбда-выражении содержится код, изменяющий заголовок формы на информацию о количестве «входов» мыши на форму

```
public Form1() {
   InitializeComponent();
   this.OnMouseEnter += (object sender, EventArgs e) => {
               this.Text = ("My Form " + counter++);
   };
}
```

Задание к задаче 3

1. В конструкторе **Form1()** добавить обработчик события **OnMouseClick**. Обработчик описать при помощи лямбда-выражения, при нажатии кнопки мыши фон формы изменять на красный.
2. В обработчик события **OnMouseEnter** добавить код, возвращающий фону формы значение по умолчанию **Form.DefaultBackColor**

1. В конструкторе **Form1()** добавить обработчик события **OnMouseDoubleClick**. Обработчик описать при помощи лямбда-выражения, при двойном нажатии форма изменяет размер на произвольный (измерения - случайные числа из диапазона от 100 до 1000).

Задача 4

Подготовительная работа:

Скопируйте и отладьте классы `UIString` и `ConsoleUI` и код консольного приложения, в котором используются объекты этих классов.

- **Объект класса `UIString` представляет строку, которую должен читать и изменять объект класса `ConsoleUI`.**
- **Объект класса `ConsoleUI` «взаимодействует» с пользователем: выводит текст (строку) из объекта класса `UIString`, получает от пользователя новый текст и заменяет этим текстом строку в объекте класса `UIString`.**

Задача 4

```
public class UIString {
    string str = "Default text";
    public string Str { get { return str; } set { str = value; } }
}
class ConsoleUI {
    UIString s = new UIString(); // специальная строка
    public UIString S { get { return s; } set { s = value; } }
    public void GetStringFromUI() {
        Console.WriteLine("Введите новое значение строки");
        string str = Console.ReadLine();
        RefreshUI();
    }
    public void CreateUI() {
        RefreshUI();
    }
    public void RefreshUI() {      // обновление строки
        Console.Clear();
        Console.WriteLine("Текст строки: " + s.Str);
    }
}
```

Задача 4

```
class Program {
  static void Main(string[] args) {
     ConsoleUI c = new ConsoleUI();
     c.CreateUI(); // запускаем выполнение объекта класса ConsoleUI
     do {
         c.GetStringFromUI();  // изменяем значение строки
         Console.WriteLine("Чтобы закончить эксперименты, нажмите ESC...");
     } while (Console.ReadKey(true).Key != ConsoleKey.Escape);
   }
}
```

Задача 4

Результатом работы метода GetStringFromUI() должно стать изменение значения поля str объекта класса UIString.

Для достижения данного результата заменить существующий код, добавив него событие «Изменение строки интерфейса» и его обработчик.

- **Добавьте в программу объявление событийного делегата-типа:**

  public delegate void NewStringValue(string s);

  В классе ConsoleUI объявите нестатическое событие NewStringValueHappened с типом делегата NewStringValue.

- **В класс UIString добавьте метод с заголовком:**

  public void NewStringValueHappenedHandler(string s).

  Метод будет обработчиком события NewStringValueHappened. В ответ на событие метод изменит значение поля str класса UIString на значение s.

  В коде метода CreateUI класса ConsoleUI добавьте в список обработчиков события NewStringValueHappened:

  метод NewStringValueHappenedHandler (тем самым объект класса UIString подпишите на получение события NewStringValueHappened).

- **Поместите в метод GetStringFromUI класса ConsoleUI код запуска (генерации) события NewStringValueHappened.**
- **Запустите и отладьте программу.**

Задача 5

Описать класс, содержащий статические методы работы с двумерным целочисленным массивом.

В классе описать метод построчной печати элементов двумерного массива (по пять элементов в строке). Перевод строки организовать при помощи событий.

Задача 5

```
using System;
public delegate void LineCompleteEvent();

public class Methods {
    // статическое событие
    public static event LineCompleteEvent lineComplete;

    public static void ArrayPrint(int[,] arr) {
        for (int i = 0; i <= arr.GetUpperBound(0); i++) {
            for (int j = 0; j <= arr.GetUpperBound(1); j++)
                Console.Write(arr[i, j]+ " " );
            lineComplete(); // событие!!
        }
    }
}
```

Задача 5

```
class Program {
    static void Main() {
        int[,] arr = new int[15, 15];
        Random rnd = new Random();
        for (int i = 0; i <= arr.GetUpperBound(0); i++)
            for (int j = 0; j <= arr.GetUpperBound(1); j++)
                arr[i, j] = rnd.Next(100);
        // в качестве обработчика - лямбда-выражение
        Methods.lineComplete += () => { Console.WriteLine(); };
        Methods.ArrayPrint(arr);
    }
}
```

Задача 6

Дополнить класс из задачи 5 кодом метода построчного заполнения двумерного массива.

Описать новое событие newItemFilled, запускаемое при присваивании нового значения очередному элементу массива. В обработчике вычислять и выводить на печать сумму элементов двумерного массива

Задача 6

```
using System;
public delegate void MethodsEvents();
public delegate void ItemEvents(int[,] ar);
public class Methods {
    public static event MethodsEvents lineComplete; //  строка заполнена
    // новый элемент проинициализирован
    public static event ItemEvents newItemFilled;
    // метод заполнения массива
    public static void ArrayFill(int[,] arr) {
        Random rnd = new Random();
        for (int i = 0; i <= arr.GetUpperBound(0); i++)
            for (int j = 0; j <= arr.GetUpperBound(1); j++) {
                arr[i, j] = rnd.Next(100);
                newItemFilled(arr);
            }
    }
    // обработчик события добавления элемента вычисляет сумму элементов
    public static void ArraySumCount(int[,] arr) {
        int sum = 0;
        for (int i = 0; i <= arr.GetUpperBound(0); i++) {
            for (int j = 0; j <= arr.GetUpperBound(1); j++)
                sum += arr[i, j];
        }
        Console.WriteLine(sum);
    }
```

Задача 6

Метод печати двумерного массива оставляем без изменений.

```
int[,] arr = new int[15, 15];
int x = 0;
Methods.newItemFilled += Methods.ArraySumCount;
Methods.ArrayFill(arr);
```

TODO: Связать с событием newItemFilled два обработчика.

- **Первый пересчитывает среднее значение всех заполненных элементов массива и выводит его на экран.**
- **Второй переопределяет значение максимального элемента среди получивших значение.**

Задача 7

Программа оценивает "трудоемкость" алгоритма сортировки массива.
В конце каждой итерации внешнего цикла сортировки возникает событие.
На событие реагируют два объекта-наблюдателя.

- **Один выводит текущее значение счетчика выполненных обменов при сортировке.**
- **Второй - визуализирует с помощью "индикатора процесса" выполнение сортировки.**

Один из методов наблюдения статический, второй - метод объекта.

Задача 7

```
using System;
// Объявление делегата-типа:
public delegate void SortHandler(long cn, int si, int kl);
class Sorting   { // класс сортировки массивов
    int[] ar;   // ссылка на массив
    public long count; // счетчик выполненных обменов при сортировке
    public event SortHandler onSort; // объявление события
    public Sorting(int[] ls)   { // конструктор
        count = 0; ar = ls;
    }
    public void Sort()   {// сортировка с посылкой извещений
        int temp;
        for (int i = 0; i < ar.Length - 1; i++) {
            for (int j = i + 1; j < ar.Length; j++)
                if (ar[i] > ar[j]) {
                    temp = ar[i];
                    ar[i] = ar[j];
                    ar[j] = temp;
                    count++;
                }
            if (onSort != null) // сортировка не завершена
                onSort(count, ar.Length, i); // генерация события
        }
    }
}
```

Задача 7

```
class View {    // Обработчик событий в объектах
    public void nShow(long n, int si, int kl) {
        Console.Write("\r" + n); // статус сортировки
    }
}
class Display { // Обработчик событий в этом классе
  static int len = 30;
  static string st = null;

  public static void BarShow(long n, int si, int kl) {
    int pos = Math.Abs((int)((double)kl / si * len));
    string s1 = new string('\u258c', pos); //код для вертикального бара
    string s2 = new string('-', len - pos - 1) +
                                  '\u25c4';   // unicode для треугольника;
        st = s1 + '\u258c' + s2; //'\u258c' - код прямоугольника
        Console.Write("\r\t\t" + st);
    }
}
```

Задача 7

```
class Controller {
    static void Main() {
        Random ran = new Random(55);
        int[] ar = new int[19999];
        for (int i = 0; i < ar.Length; i++)
            ar[i] = ran.Next();
        Sorting run = new Sorting(ar);
        View watch = new View();    // Создан объект
        run.onSort += new SortHandler(Display.BarShow);
        run.onSort += new SortHandler(watch.nShow);
        run.Sort();
        Console.Write("\n");
    }
}
```

Задачи для самостоятельного решения

- *В библиотеке классов описать:*
  - событийный делегат **CoordChanged**, представляющий методы одним параметром типа **Dot**, возвращающие значение **void**.
  - класс **Dot** – точка на плоскости с двумя вещественными координатами. В классе разместите:
  - событие **OnCoordChanged** с типом **CoordChanged**;
  - метод **DotFlow()**, который в цикле присваивает координатам точки **10** раз случайные вещественные значения из диапазона **(-10; 10)**. Если обе координаты точки отрицательные значения – запускать событие **OnCoordChanged**, и передавать в него ссылку на объект.
- *В основной программе:*
  - описать статический метод **PrintInfo(A)**, возвращающий значение **void** и выводящий в консоль координаты объекта **A** типа **Dot**, переданного в качестве параметра.
  - В методе **Main()** получить от пользователя две вещественные координаты **X** и **Y**. Создать объект **D** типа **Dot** с координатами **X** и **Y**. Подписать метод **PrintInfo()** на событие **OnCoordChanged**. Запустить для объекта **D** метод **DotFlow()**.

Задачи для самостоятельного решения

- *В библиотеке классов описать:*
  - событийный делегат **SquareSizeChanged**, представляющий методы с одним вещественным параметром и возвращающие значение **void**.
  - класс **Square** – квадрат на плоскости. Квадрат задан координатами верхнего левого и правого нижнего углов. В классе разместите:
  - событие **OnSizeChanged** с типом **SquareSizeChanged**;
  - свойства доступа к координатам определяющих углов квадрата. В коде каждого свойства после изменения значения поля запускает событие **OnSizeChanged**, в качестве параметра передаётся новое значение длины стороны квадрата.
- *В основной программе:*
  - описать статический метод **SquareConsoleInfo(A)**, возвращающий значение **void** и выводящий в консоль, с точностью до двух знаков после запятой вещественное число **A**, переданное в качестве параметра.
  - В методе **Main()** получить от пользователя координаты углов квадрата. На основе этих координат создать объект **S** типа **Square**. Связать метод **SquareConsoleInfo()** с событием **OnSizeChanged** объекта **S**. В цикле получать от пользователя координаты правого нижнего угла квадрата **X** и **Y**, используя свойства объекта **Square**, изменять координаты углов **S**, условие окончания цикла определить самостоятельно.