

# Дополнительные возможности классов .NET

---

- Делегаты
- События
- Создание пользовательских событий

# Модель разработки приложений в .NET

---

- Элементы управления: `CheckedListBox`, `PictureBox`, `RadioButton`, `GroupBox`, `Panel`. Их свойства.

# Делегат

---

ы

**Делегат** – это тип, который представляет ссылки на методы с определенным списком параметров и типом возвращаемого значения.

**Делегат** – это тип, который безопасно инкапсулирует метод, схожий с указателем функции в С и С++.

В отличие от указателей функций в С делегаты объектно-ориентированы, типобезопасны и безопасны.

# Делегат

---

ы

## Использование делегатов:

1. Объявляем делегат
2. Создаем экземпляр делегата
3. Регистрируем метод в делегате
4. Регистрируем другие методы (при необходимости)
5. Исключаем методы из делегата(при необходимости)
6. Вызываем делегат

# Делегат

---

ы

```
class Библиотека
{
    public void Book1()
    {
        Console.WriteLine("Война и мир");
    }

    public void Book2()
    {
        Console.WriteLine("Братья Карамазовы");
    }
}
```

# Делегат

---

```
class Program
{
    public delegate void MyBook(); //объявляем делегат

    static void Main(string[] args)
    {
        //создаем экземпляр делегата и регистрируем
        МЕТОД
        MyBook myBook = new MyBook(new Biblioteka().Book1);

        //добавляем методы
        myBook += new MyBook(new Biblioteka().Book2);

        myBook(); //вызываем делегат
    }
}
```

# Делегат

---

ы

## Для чего нужны делегаты?

1. Обеспечение инкапсуляции
2. В случае многоадресного делегата: вызов нескольких методов одним вызовом делегата
3. Синхронный и асинхронный вызов метода
4. Основа события

# Событи

---

Я

**Событие** представляет собой сообщение, посылаемое объектом, чтобы сигнализировать о совершении какого-либо действия.

**Событие** означает, что произошло или вскоре произойдет изменение в состоянии объекта.

Обработчик события – делегат, на котором событие основано.

# События

---

## Я

**static** – Делает событие доступным для вызова в любое время, даже если экземпляр класса отсутствует.

**virtual** – Позволяет производным классам переопределять поведение события при помощи ключевого слова `override`.

**sealed** – Указывает, что для производных классов событие более не является виртуальным.

**abstract** – Компилятор не создаст блоки методов доступа к событиям `add` и `remove` и, следовательно, производные классы должны предоставлять собственную реализацию.



# События

---

Я

## Объявление события:

```
public delegate void MyDelegate(string str);  
public event MyDelegate MyEvent;
```

При разработке обычно используют делегаты **EventHandler** – встроенные обработчики событий .NET

```
public delegate void EventHandler(object sender,  
EventArgs e);
```

# События

---

Я

**Обработчик события** – это делегат со специальной сигнатурой.

```
public delegate void MyEventHandler(object sender,  
MyEventArgs e);
```

**sender** – определяет объект, который издает событие.

**e** – содержит данные, которые должны быть использованы обработчиком события.

# События

---

Я

**Событие** – это именованный делегат, при вызове которого, будут запущены все подписавшиеся на момент вызова события методы заданной сигнатуры.

event delegateName **eventName**

# События

---

Я

```
delegate void MyDlg ();
```

```
class Class1
```

```
{
```

```
    public event MyDlg MyEvent;
```

```
    public void EventInit ()
```

```
    {
```

```
        if (MyEvent != null) MyEvent ();
```

```
    }
```

```
}
```

# События

---

```
class Program
{
    static void Handler()
    {
        Console.WriteLine("Произошло событие");
    }

    static void Main(string[] args)
    {
        Class1 cl1 = new Class1();
        cl1.MyEvent += new MyDlg(Handler);

        cl1.EventInit();

        Console.ReadLine();
    }
}
```

# Событі

---

я

Событія могут активизировать несколько обработчиков. Такие события называются **широковещательными**.

# События

---

## Я

```
class C11
{
    public void HandlerC1 ()
    { Console.WriteLine ("Событие получено объектом класса C11"); }
}
class C12
{
    public void HandlerC2 ()
    { Console.WriteLine ("Событие получено объектом класса C12"); }
}
class C13
{
    public void HandlerC3 ()
    { Console.WriteLine ("Событие получено объектом класса C13"); }
}
```

# События

---

```
static void Handler()
{ Console.WriteLine("Событие получено объектом класса Program"); }

static void Main(string[] args)
{
    Class1 cl = new Class1();
    cl.MyEvent += new MyDlg(Handler);

    C11 c11 = new C11();
    cl.MyEvent += new MyDlg(c11.HandlerC1);

    C12 c12 = new C12();
    cl.MyEvent += new MyDlg(c12.HandlerC2);

    C13 c13 = new C13();
    cl.MyEvent += new MyDlg(c13.HandlerC3);

    cl.EventInit();

    Console.WriteLine("Удаляем одно событие:");
    cl.MyEvent -= new MyDlg(c12.HandlerC2);
    cl.EventInit();
}
```



# Событи

---

Я

## Создание событий:

1. Определить условие возникновения события и методы, которые должны сработать.
2. Создать делегат на основе сигнатуры этих методов.
3. Создать событие на основе этого делегата и вызвать его когда условие сработает.
4. Подписаться на событие методами, которые должны сработать.

# Элементы

---

## управления

**CheckedListBox** – содержит коллекцию отмеченных и неотмеченных флажками элементов в списке

### **Свойства:**

**Count** – возвращает кол-во элементов списка.

**Item[]** – возвращает один из элементов **CheckedListBox**.

# Элементы управления

---

**Добавление элемента в коллекцию:**

```
checkedListBox1.Items.Add("строка", true);
```

**4-ый элемент коллекции:**

```
checkedListBox1.Items[3];
```

**1-ий из выбранных элементов**

**коллекции:**

```
checkedListBox1.CheckedItems[0]
```

# Элементы управления

---

**PictureBox** – элемент управления графическим окном Windows для отображения рисунка

## **Свойства:**

**Image** – возвращает или задает изображение,

отображаемое в PictureBox

**SizeMode** – указывает, как изображение отображается

# Элементы

---

## управления

**RadioButton** – возвращает или задает значение, указывающее, отмечен ли элемент управления.

```
if (radioButton1.Checked)
```

```
    MessageBox.Show("Элемент выбран");
```

**GroupBox** – элемент управления, создающий контейнер с границей и заголовком.

# Лабораторная

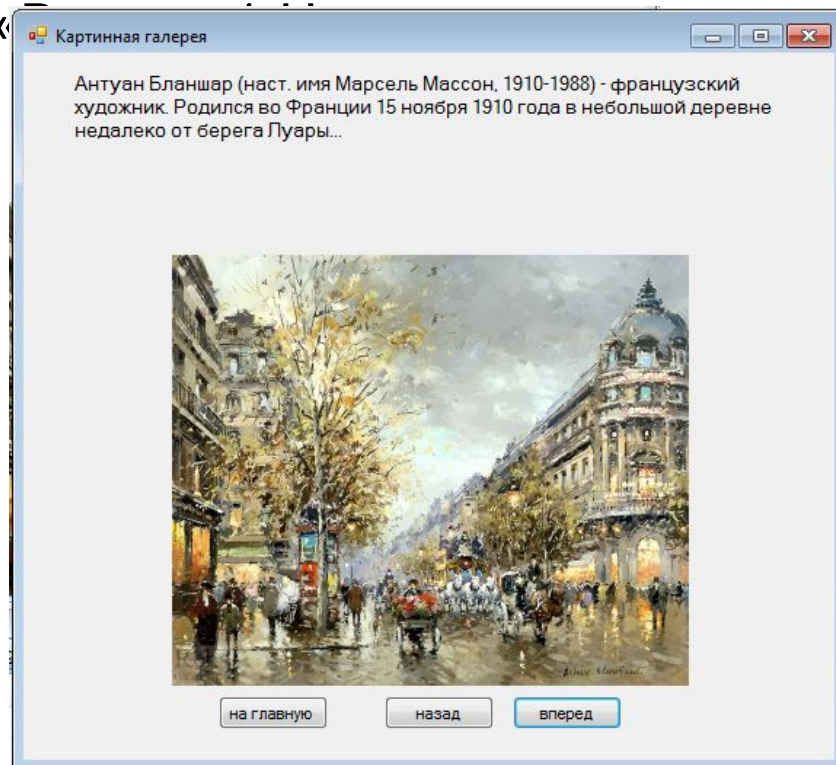
## работа 6

Проект должен состоять из двух окон (форм): «Главная» и «Просмотр».

1 форма должна содержать список художников.

На 2 форме – необходимо разместить краткую биографию выбранного художника и предоставить возможность просмотра его картин (элемент PictureBox).

Добавьте кнопки «назад» и «вперед» для переключения изображений.



ания