ОБРАЗОВАТЕЛЬНЫЙ ЦЕНТР
ПРОГРАММИРОВАНИЯ И ВЫСОКИХ
ТЕХНОЛОГИЙ

2020г.

Специальные типы классов

AGENDA

- ✓ Абстрактный класс
- ✓ Статический класс
- ✓ Встроенный класс (nested)
- ✓ Разделяемый класс

Абстрактный класс

Абстрактный клас – это класс, в котором хотя бы один из следующих элементов объявлен абстрактным: Метод, Свойство, Индексатор, Событие.

Используется ключевое слово **abstract**.

Объект(экземпляр) абстрактного класса создать нельзя.

Используются при организации наследования.

Абстрактные методы должны быть обязательно переопределены в производном классе.

Переопределение при помощи ключевого слова **override**.

Абстрактные члены классов не должны иметь модификатор **private**.

Абстрактный класс

Абстрактный класс может применять определенный шаблон проектирования для классов, которые его расширяют.

Пример использования:

В классе существует метод, реализация которого для каждого дочернего класса будет отличаться, но метод должен быть там, тогда этот метод может быть объявлен как абстрактный в абстрактном классе и в то же время общие методы, реализация которых остается такой же, могут быть унаследованы дочерним классом от абстрактного родительского класса.

```
// абстрактный класс фигуры
abstract class Figure
{
    // абстрактный метод для получения периметра
    public abstract float Perimeter();

    // абстрактный метод для получения площади
    public abstract float Area();
}
```

Статический класс

Статические классы объявляются с модификатором **static** и могут содержать только статические поля, свойства и методы.

Статические классы имеют следующие ограничения:

- Нельзя создавать экземпляр класса, используя ключевое слово **new**.
- Не разрешается использовать не статические члены этого же класса.
- Он не поддерживает наследование.

Могут иметь статический конструктор, который:

- Не должен иметь модификатор доступа и не принимает параметров.
- Нельзя использовать ключевое слово **this** для ссылки на текущий объект класса и можно обращаться только к статическим членам класса.
- Нельзя вызвать в программе вручную. Они выполняются автоматически при самом первом создании объекта данного класса или при первом обращении к его статическим членам.

Статические конструкторы обычно используются для инициализации статических данных, либо же выполняют действия, которые требуется выполнить только один раз.

Нестатические классы также должен определять статический конструктор, если класс содержит статические члены, для которых нужна нетривиальная инициализация.

Статический класс

Если класс не является статическим, но содержит статические методы, методы имеют следующие ограничения:

- Не разрешается использовать не статические члены этого же класса из статических.
- Наследование и полиморфизм для статических членов не поддерживаются.

```
// статический класс
static class Math
{
    // абстрактный метод возведения в квадрат
    public static double Sqrt(double number)
    {
        return number * number;
    }
}
```

Встроенный класс

Встроенный(вложенный, nested) класс объявляется внутри области видимости другого типа.

Если единственной причиной для использования вложенного типа является желание избежать загромождения пространства имен слишком большим числом типов, рассмотрите возможность применения вместо этого вложенного пространства имен. Вложенный тип должен использоваться из-за его более строгих ограничений контроля доступа или же когда вложенному классу нужен доступ к закрытым членам включающего класса.

```
public class TopLevel
{
    public class Nested // Вложенный класс
    {
        public enum Color // Вложенное перечисление
        {
            Red, Blue, Tan
        }
    }
}
```

Встроенный класс

Встроенный класс обладает следующими характеристиками:

Он может получать доступ к закрытым членам включающего типа и ко всему остальному, к чему имеет доступ включающий тип.

Он может быть объявлен с полным диапазоном модификаторов доступа, а не только с **public** и **internal**.

Стандартной доступностью вложенного типа является **private**, а не **internal**.

Доступ к вложенному типу извне требует указания имени включающего типа (как при обращении к статическим членам).

Разделяемый класс

Разделяемые классы (partial class) позволяют классам, структурам и интерфейсам быть разбитыми на несколько фрагментов, описанных в отдельных файлах с исходным текстом. При компиляции все эти определения будут скомпилированы в одно.

```
//Class1.cs
partial class MyClass
{
    private int myField;
}
```

```
//Class2.cs
partial class MyClass
{
    public int GetField()
    {
        return myField;
    }
}
```

Использование большого класса разделенного на несколько файлов, чтобы облегчить работу с различными частями класса.

При работе с автоматически сгенерированным кодом в одном файле хранится сгенерованный код, а в другом – добавленный программистом.

Разделяемый класс

Разделяемый метод состоит из двух частей: заголовка и реализации.

Разделяемые методы подчиняются следующим правилам:

- Объявление метода возможно только в **partial-классе**.
- Объявление метода начинается с модификатора **partial**.
- Метод обязан возвращать значение **void**.
- Метод может иметь параметры, но **out**-параметры запрещены.
- Метод неявно объявляется как **private**. Поэтому он не может быть виртуальным.
- Разделяемые методы могут быть статическими или универсальными.
- Реализация может быть опущена.

```
//Class1.cs
partial class MyClass
{
    partial void DoSomethingElse();
}
```

```
//Class2.cs
partial class MyClass
{
    partial void DoSomethingElse()
    {

Console.WriteLine();
    }
}
```

Методы расширения

Методы расширения (**extension methods**) позволяют добавлять новые методы в уже существующие типы без создания нового производного класса.

Метод расширения, как и класс его содержащий должный быть статичесими **static**.

В качестве первого параметра метод должен принимать тот тип для которого он будет вызываться с ключевым словом **this**.

```
public static class StringExtension
{

    public static int CharCount(this string str)
    {
        return str.Length();
    }
}

// Применение
string str = “hello world!”;
int count = str.CharCount(); // count = 12
```

Задание

1. Создайте проект по шаблону Console Application и ClassLibrary с типом Rectangle.

В теле класса создать два поля, описывающие длины сторон double side1, double side2.

Создать пользовательский конструктор Rectangle(double side1, double side2), в теле которого поля side1 и side2 инициализируются значениями аргументов.

Создать два **private** метода, вычисляющие площадь прямоугольника - double AreaCalculator() и периметр прямоугольника - double PerimeterCalculator().

Создать два свойства double Area и double Perimeter с одним методом доступа get.

Написать программу, которая принимает от пользователя длины двух сторон прямоугольника и выводит на экран периметр и площадь.

2. Создайте метод расширения для массива int, который возвращает наибольший элемент.

3. Создайте тип Book и вложенный в него тип Notes. Book должен позволять добавлять заметки в себя (привязывается к странице) и читать все заметки из себя (страница -> текст).