Лабораторная работа №1: Отображение геометрических фигур

Цель работы:

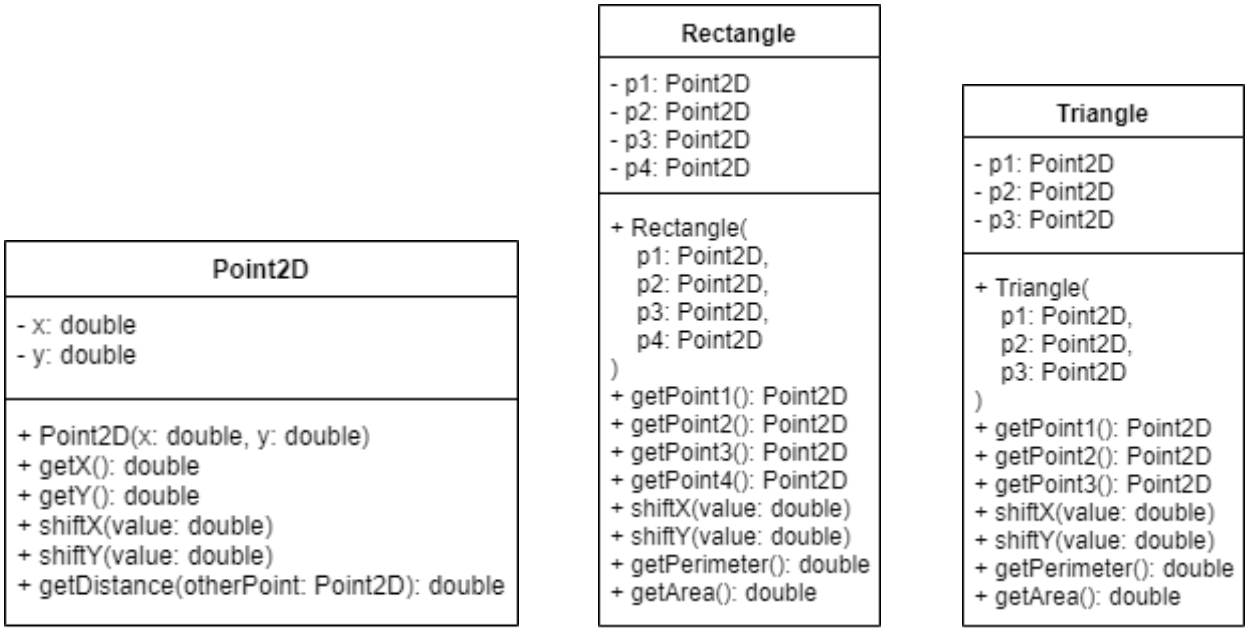
- изучение способов отображения геометрических фигур на языке C#;
- знакомство с базовыми принципами ООП и терминами «класс» и «объект».

Задание:

В рамках выполнения лабораторной работы необходимо реализовать приложение WPF, содержащее несколько классов геометрических фигур на плоскости:

1) Класс «двумерная точка». Класс должен содержать поля для хранения координат по осям X и Y и методы, реализующие следующие операции:
- сдвиг точки по осям X и Y на заданное расстояние;
- вычисление расстояния между двумя точками;
2) Класс «треугольник». Класс должен содержать поля для хранения вершин треугольника и методы, реализующие следующие операции:
- вычисление площади и периметра;
- сдвиг треугольника по осям X и Y на заданное расстояние;
3) Класс «прямоугольник». Класс должен содержать поля для хранения вершин прямоугольника и методы, реализующие следующие операции:
- вычисление площади и периметра;
- сдвиг прямоугольника по осям X и Y на заданное расстояние;
4) Класс для генерации геометрических фигур. Класс должен содержать статические методы создания геометрических фигур:
- создание произвольной («рандомной») точки;
- создание произвольного треугольника;
- создание произвольного прямоугольника;
- создание прямоугольника заданного размера.

Ниже изображены классы геометрических фигур в виде UML-блоков:



Конструкторы классов принимают набор параметров для инициализации (заполнения) полей класса.

Интерфейс программы должен представлять собой несколько пунктов меню (или кнопок), соответствующих одной из геометрических фигур – «Произвольный треугольник», «Произвольный прямоугольник», «Квадрат» и др., по нажатию на которые осуществляется рисование фигуры в окне и вывод информации (координат вершин, периметра, площади). Также должны присутствовать

пункты меню для сдвига фигур, по нажатию на которые фигура должна сдвигаться и перерисовываться.

Краткие теоретические сведения:

Класс – это описание (шаблон) набора объектов с одинаковыми атрибутами, операциями, связями и семантикой.

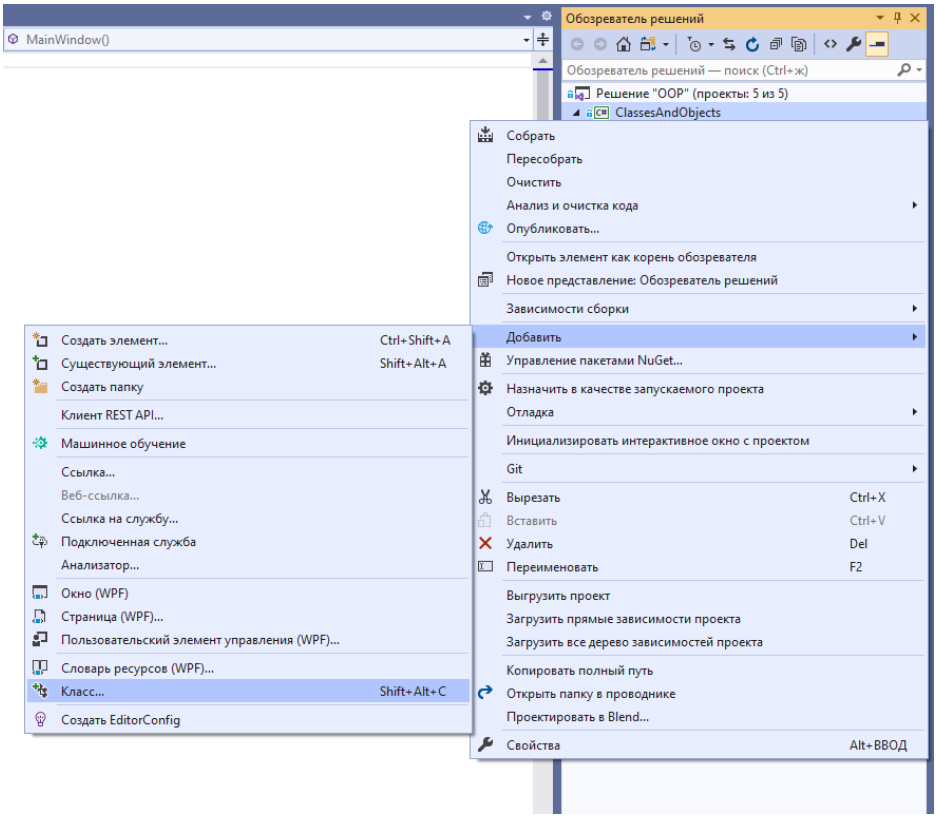
Класс является логической абстракцией. Физическое представление класса появится в оперативной памяти лишь после того, как будет создан **объект** (экземпляр) этого класса.

На языке программирования C# класс создается с помощью ключевого слова **class**. Ниже приведена общая структура класса, содержащая несколько полей (в терминах UML – атрибуты) и методов (в терминах UML – операции).

```
class имяКласса
{
    // объявление полей класса
    доступ тип переменная1;
    доступ тип переменная2;
    //...
    доступ тип переменнаяN;

    // объявление методов класса
    доступ возвращаемыйТип метод1(параметры)
    {
        // тело метода
    }

    доступ возвращаемыйТип метод2(параметры)
    {
        // тело метода
    }

    // ...

    доступ возвращаемыйТип методN(параметры)
    {
        // тело метода
    }
}
```

Перед каждым объявлением поля или метода указывается модификатор доступа. **Модификатор доступа** определяет тип разрешенного доступа. Указывать модификатор доступа не обязательно, но если он отсутствует, то объявляемый член считается закрытым в пределах класса. Члены с закрытым доступом **private** могут использоваться только другими членами их класса. Члены класса с типом доступа **public** доступны везде за пределами данного класса.

Добавление класса осуществляется в контекстном меню проекта: необходимо выбрать пункт «Добавить – Класс...» (или воспользоваться комбинацией клавиш Shift+Alt+C). В появившемся диалоговом окне необходимо ввести имя файла. В целях упрощения структуры проекта рекомендуется придерживаться принципа «один файл – один класс».



На основе приведенной выше общей структуры класса можно определить класс Date:

```
// класс даты
class Date
{
    private int day; // день
    private int month; // месяц
    private int year; // год

    // инициализация
    public void Init(int d, int m, int y)
    {
        day = d;
        month = m;
        year = y;
    }

    // узнать день
    public int GetDay()
    {
        return day;
    }

    // узнать месяц
    public int GetMonth()
    {
        return month;
    }

    // узнать год
    public int GetYear()
    {
        return year;
    }
}
```

В сущности, класс – это пользовательский тип данных. Чтобы начать работу, нужно создать объект (экземпляр) этого класса (объявить переменную этого пользовательского типа):

```
class Program
{
    static void Main(string[] args)
    {
        Date date = new Date(); // создание объекта класса с именем date
        date.Init(21, 2, 2019); // инициализация объекта (заполнение данными)

        // использование метода GetDay для вывода дня
        Console.WriteLine("Day: {0}", date.GetDay());
        Console.ReadKey();
    }
}
```

Для инициализации объекта также можно воспользоваться конструктором класса. **Конструктор** – метод, инициализирующий объект при его создании. У конструктора такое же имя, как и у его класса, а с точки зрения синтаксиса он подобен методу, за исключением явно указываемого возвращаемого типа. Конструктор класса вызывается автоматически при использовании оператора `new`.

Ниже приведена общая структура конструктора:

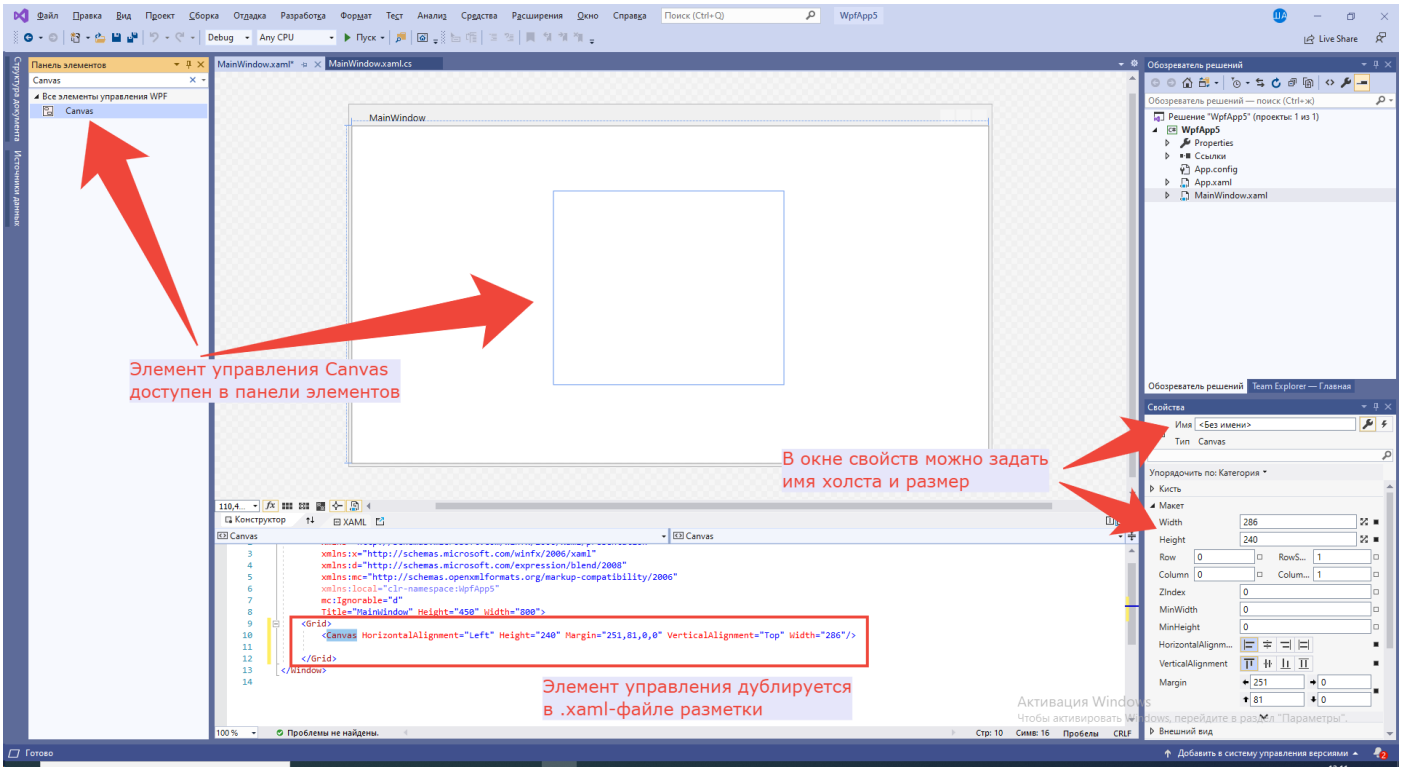
```
доступ имяКласса(параметры)
{
    // тело конструктора
}
```

Конструктор используется для инициализации полей экземпляра класса или же для выполнения любых других установочных процедур. Список параметров может быть как пустым, так и содержащим один или более параметр. Пример использования конструктора класса для инициализации его полей:

```
// класс даты
class Date
{
    private int day; // день
    private int month; // месяц
    private int year; // год

    // конструктор класса
    public Date(int d, int m, int y)
    {
        day = d;
        month = m;
        year = y;
    }
    ...
}
class Program
{
    static void Main(string[] args)
    {
        Date date = new Date(21, 2, 2019); // создание и инициализация объекта
        ...
    }
}
```

Рисование геометрических фигур на плоскости

Для рисования геометрических фигур в окне необходимо создать «холст» в области окна. В качестве «холста» можно использовать элемент управления Canvas. Добавить Canvas в область окна можно двумя способами: перетащить из панели элементов или прописать в .xaml-файле вручную:



Для рисования треугольника и прямоугольника достаточно уметь рисовать линию. Наиболее простой способ – использовать встроенный класс Line. Рисование линии с помощью класса Line состоит из следующих шагов:

- создание объекта (экземпляра) класса Line;
- задание визуальных параметров линии: цвета и толщины;
- задание относительных координат начала и конца линии;
- добавление линии в холст.

На языке программирования C# это выглядит следующим образом:

```
// создание объекта (экземпляра) класса Line
Line line = new Line();

// задание цвета линии (красный) с помощью свойства Stroke
line.Stroke = Brushes.Red;
// задание толщины линии (5) с помощью свойства StrokeThickness
line.StrokeThickness = 5;

// задание относительных координат начала и конца линии с помощью свойств X1, Y1, X2, Y2
line.X1 = 10;
line.Y1 = 50;
line.X2 = 10;
line.Y2 = 10;

// добавление линии в холст (линия добавляется в список "дочерних" элементов холста)
canvas.Children.Add(line);
```

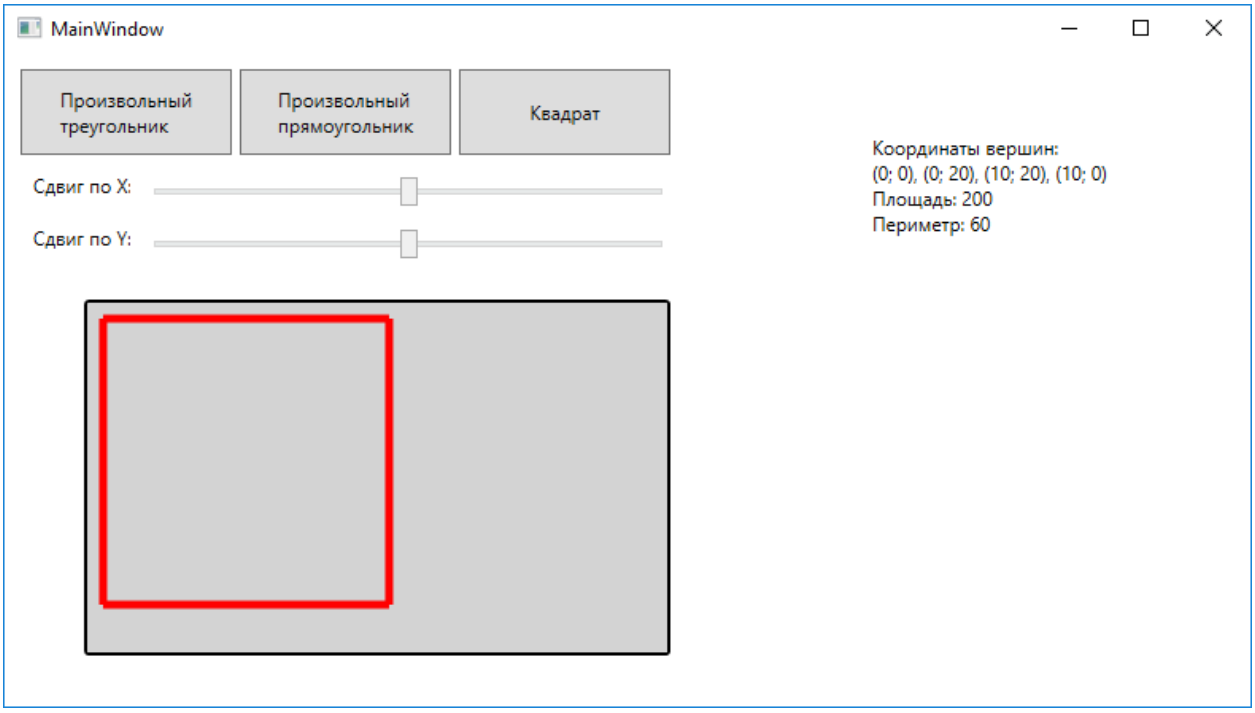
В качестве эксперимента можно добавить этот код в класс окна (по умолчанию - `MainWindow`) и проверить его работоспособность.

Для рисования треугольника необходимо нарисовать три таких линии, для рисования прямоугольника – четыре. Чтобы не дублировать код для рисования линии, необходимо данный код вынести в отдельный метод. При этом класс `MainWindow` можно организовать в соответствии с таким UML-блоком:

MainWindow
- drawLine(begin: Point2D, end: Point2D) - drawTriangle(triangle: Triangle) - drawRectangle(rectangle: Rectangle)

Операции `drawTriangle` и `drawRectangle` используют внутри себя `drawLine`.

Макет интерфейса с нарисованным прямоугольником:



Произвольное создание фигур

Для создания произвольных чисел на языке C# используется класс `Random` – генератор случайных чисел. Класс `Random` содержит метод `NextDouble()`, возвращающий случайное число от 0 до 1. Координаты точки можно сгенерировать следующим образом:

```
// создание объекта - генератора случайных чисел
Random random = new Random();
// генерация координат точки
double x = random.NextDouble();
double y = random.NextDouble();
```

Чтобы сгенерировать координаты точки в произвольных пределах, необходимо результат метода `NextDouble()` домножить на некоторую константу. Код будет выглядеть следующим образом:

```
// создание объекта - генератора случайных чисел
Random random = new Random();
// генерация координат точки в пределах [0; 100]
const int MAX = 100;
double x = random.NextDouble() * MAX;
double y = random.NextDouble() * MAX;
```

Для создания произвольного треугольника необходимо сгенерировать координаты трёх вершин – трёх пар случайных чисел. Проверкой на нахождение этих вершин не на одной прямой можно пренебречь.

Для создания произвольного прямоугольника достаточно сгенерировать координаты одной из вершин, длины двух смежных сторон и угол наклона прямоугольника. Координаты остальных вершин вычисляются относительно первой по теореме Пифагора.

Контрольные вопросы:

1) Что такое класс? Создание класса на языке программирования C#.
2) Общая структура класса, объявление полей и методов.
3) Конструктор класса.
4) Создание объекта (экземпляра класса). Доступ к полям и методам.

Список литературы:

1) Герберт Шилдт "C# 4.0: полное руководство"
2) Эндрю Троелсен "Язык программирования C# 5.0 и платформа .NET 4.5"
3) Руководство по программированию на C# https://docs.microsoft.com/ruru/dotnet/csharp/programming-guide/index
4) Полное руководство по языку программирования C# 7.0 и платформе .NET 4.7 https://metanit.com/sharp/tutorial/
5) Руководство по WPF https://metanit.com/sharp/wpf/
6) C# 5.0 и платформа .NET 4.5 http://professorweb.ru/my/csharp/charp_theory/level1/infocsharp.php
7) https://github.com/Microsoft/WPF-Samples