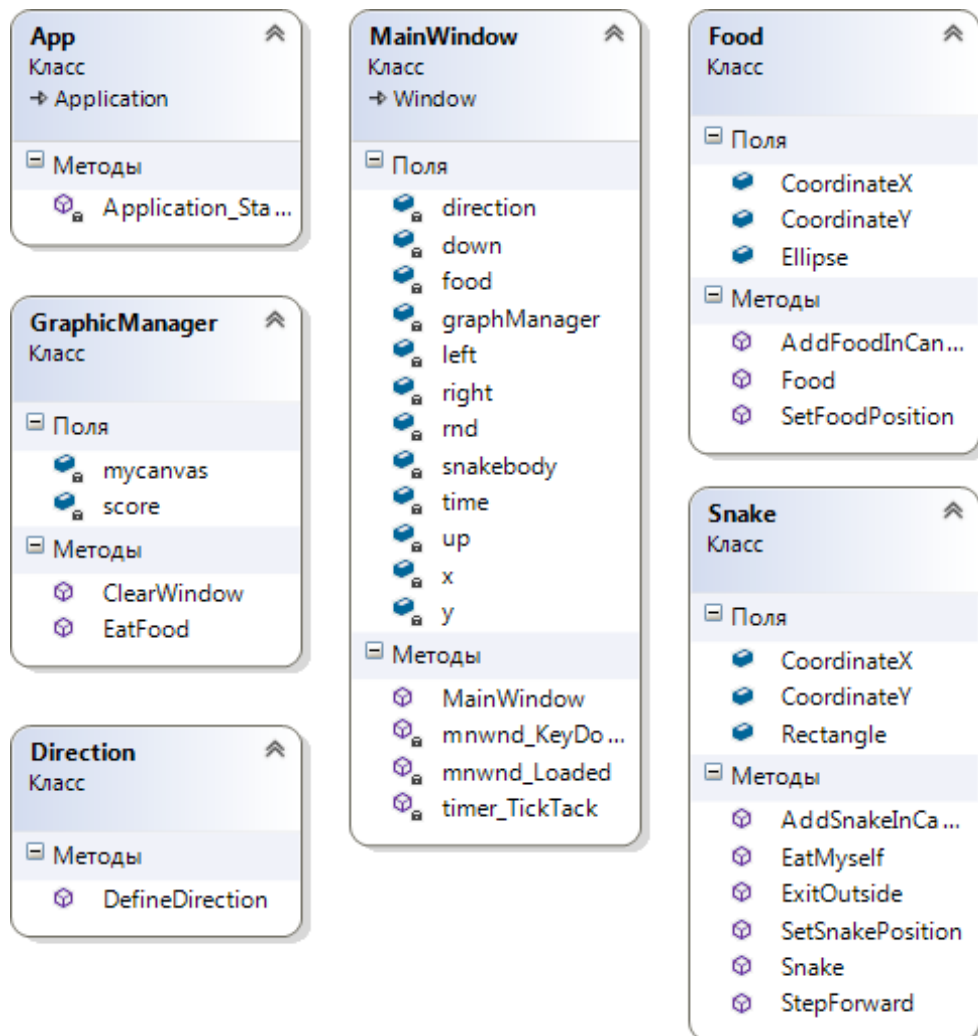


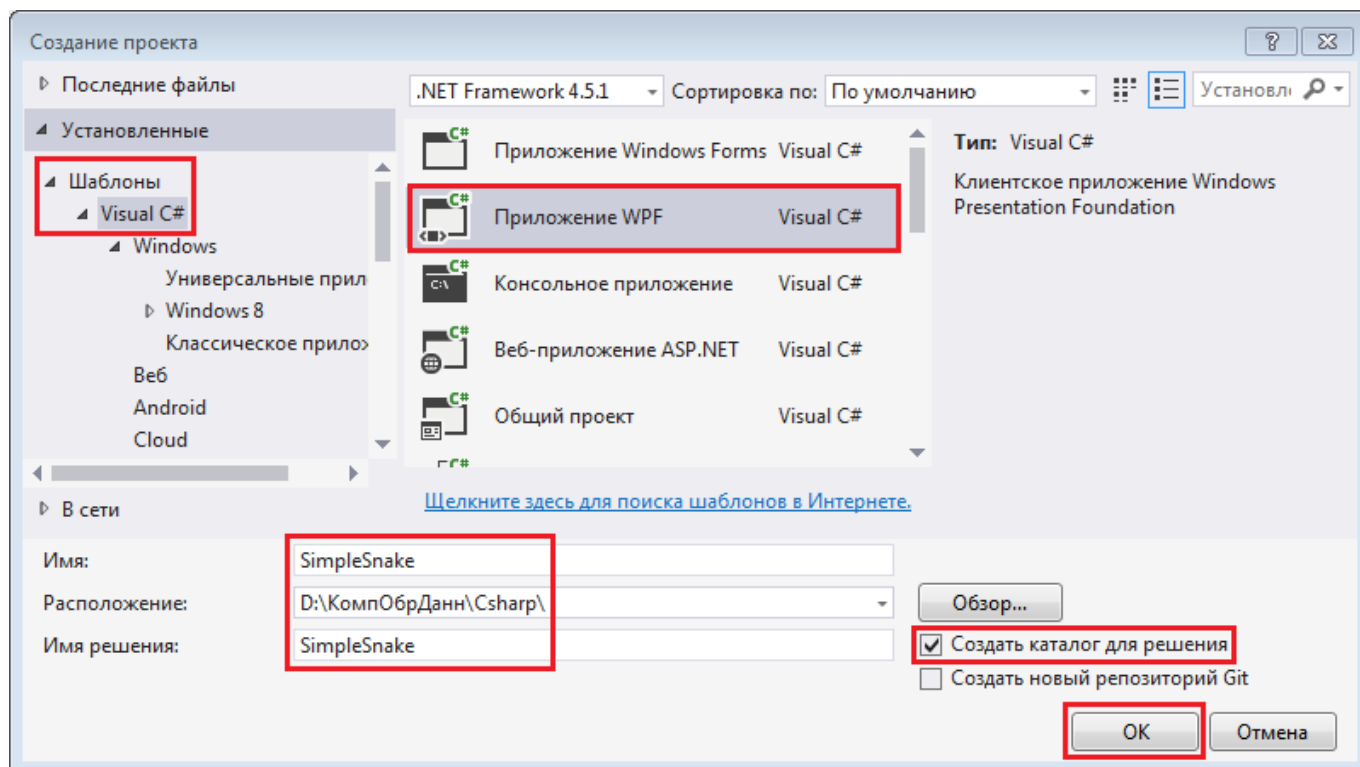
# Лабораторная работа № 4

## «Простая Змейка»

### Диаграмма классов



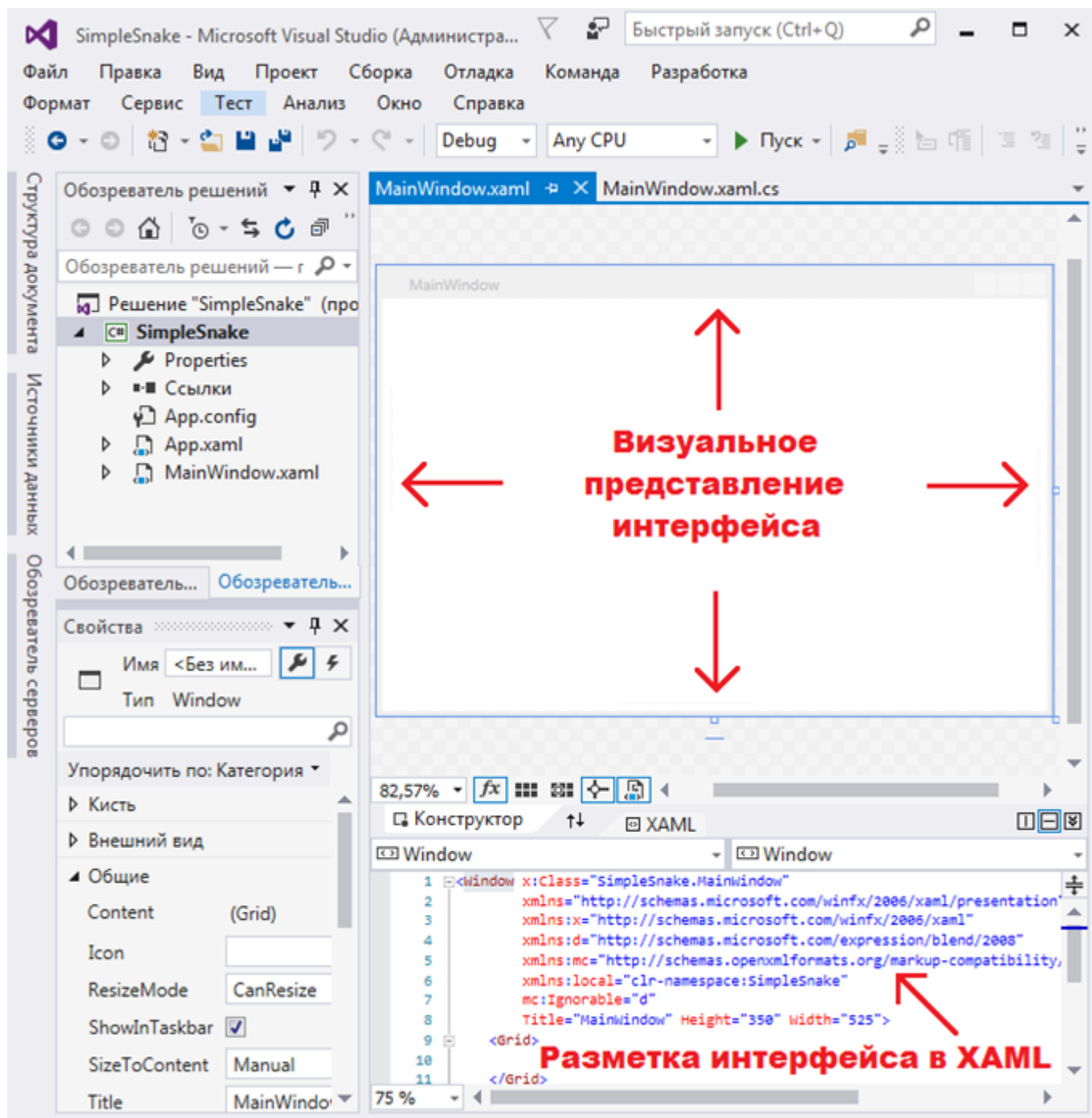
1. Создаём проект с названием **SimpleSnake** в папке **d:\КомпОбрДанн\Csharp\**.  
Шаблоны – **Visual C#**                      Шаблон – **Приложение WPF**  
Имя – **SimpleSnake**                      Расположение - **d:\КомпОбрДанн\Csharp**  
Создать каталог для решения – поставить галочку



По умолчанию **Visual Studio** создает и открывает два файла: файл декларативной разметки интерфейса **MainWindow.xaml** (конструктор для окна приложения) и файл связанного с этой разметкой кода **MainWindow.xaml.cs**. Файл **MainWindow.xaml** имеет два представления: *визуальное* – в этом режиме отображает весь графический интерфейс данного окна приложения, и под ним *декларативное* объявление интерфейса в XAML. Если изменить декларативную разметку, например, определить в разметке кнопку, то эти изменения отобразятся в визуальном представлении.

**XAML** (англ. eXtensible Application Markup Language) — расширяемый язык разметки для приложений.

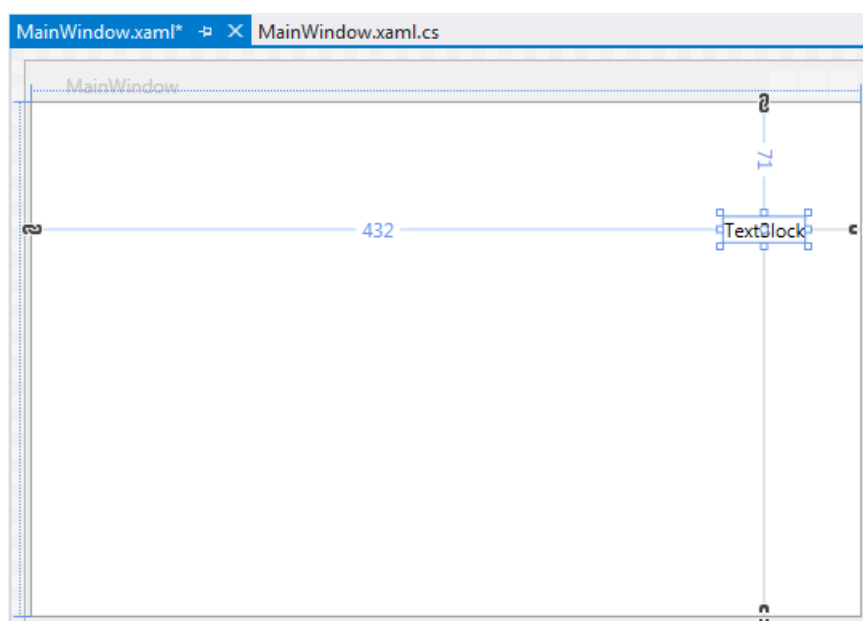
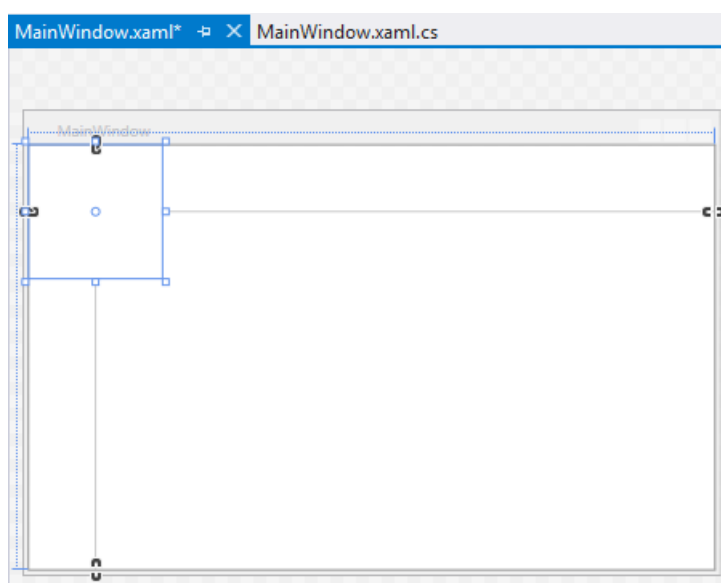
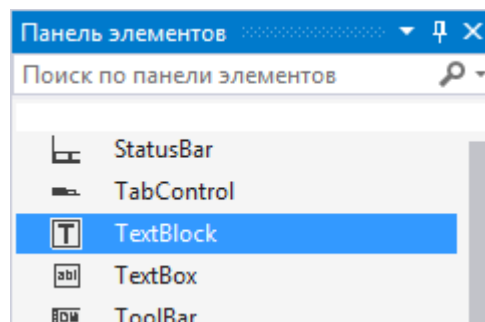
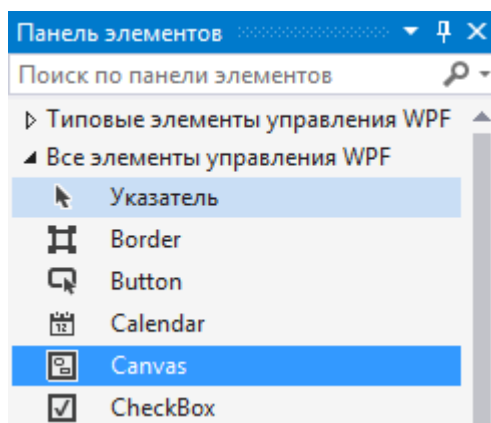
В структуре проекта WPF следует выделить следующие моменты: в проекте имеется файл **App.xaml** и связанный с ним файл кода **App.xaml.cs** – это глобальные файлы для всего приложения. **App.xaml** задает файл окна программы, которое будет открываться при запуске приложения. Если открыть этот файл, то в нем содержится строка **StartupUri="MainWindow.xaml"** – то есть в данном случае, когда запустится приложение, будет создаваться интерфейс из файла **MainWindow.xaml**.



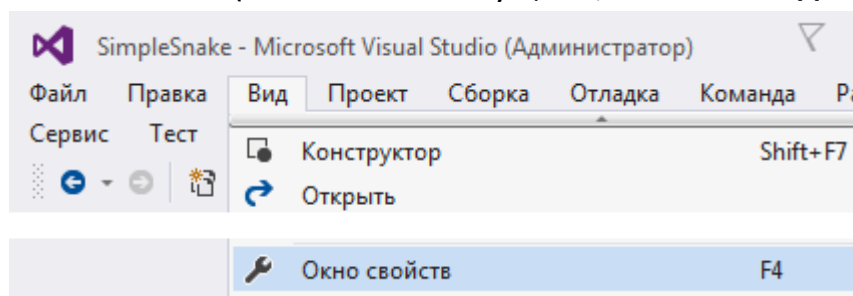
## 2. Из меню Вид запустить Панель элементов

Файл	Правка	Вид	Проект	Сборка	Отладка	Команда	Разрабо
			Код			F7	
			Конструктор			Shift+F7	
			Список задач			Ctrl+\, T	
			Панель элементов			Ctrl+Alt+X	

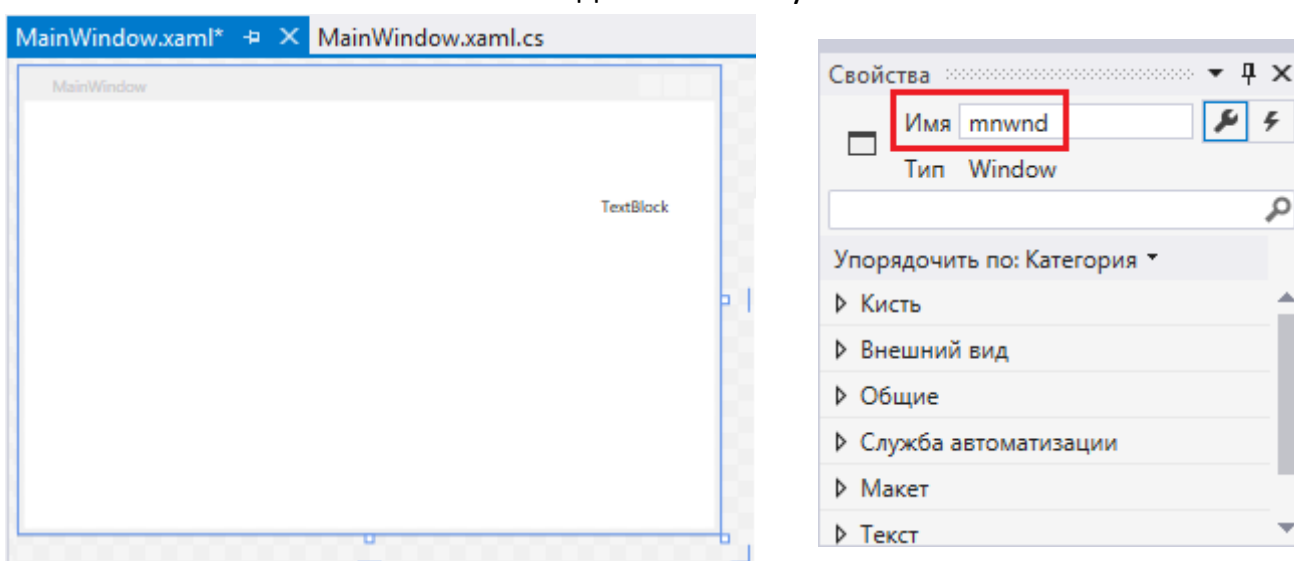
3. Поместить в конструктор окна приложения (**MainWindow.xaml**) один элемент **Canvas**, и один элемент **TextBlock**.



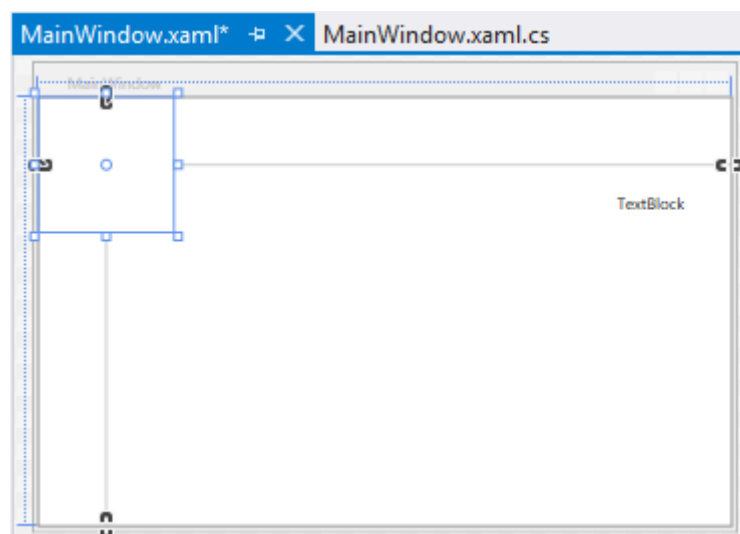
4. Перейти в окно свойств (если оно не запущено, то меню **Вид – Окно свойств**).



5. Одинарным кликом левой кнопкой мыши выделить в конструкторе окна элемент **MainWindow**. Элемент должен быть в фокусе (выделен синей линией). В окне **Свойства** в поле **Имя** задать элементу **MainWindow** имя **mnwnd**.



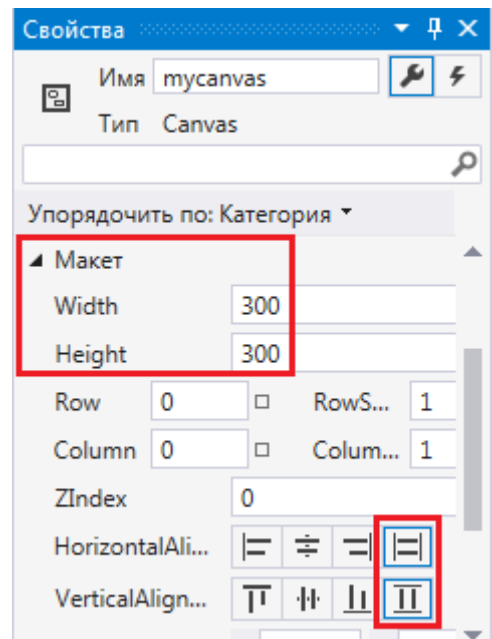
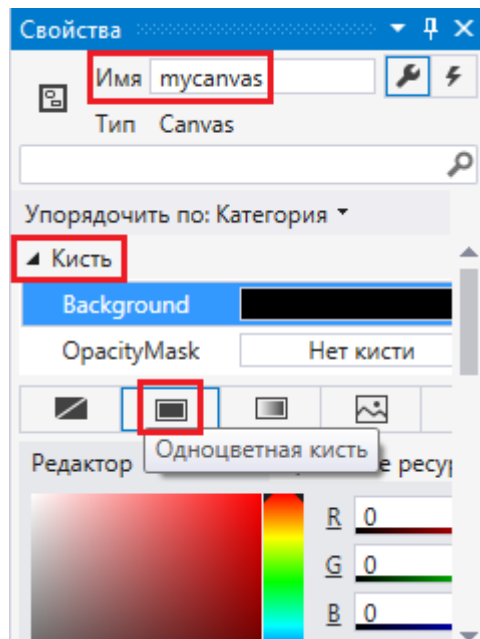
6. Одинарным кликом левой кнопкой мыши выделить в конструкторе окна элемент **Canvas**.



В окне **Свойства** в поле **Имя** задать элементу **Canvas** имя **mycanvas**.

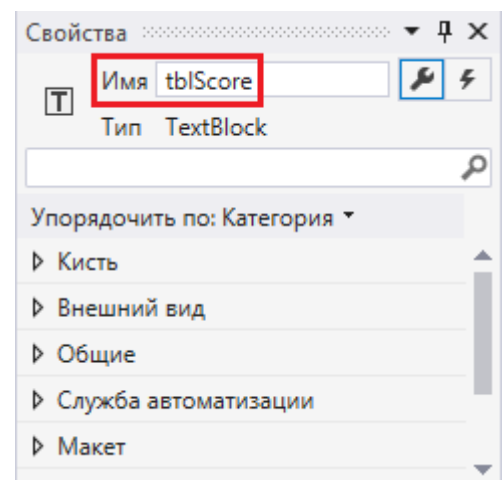
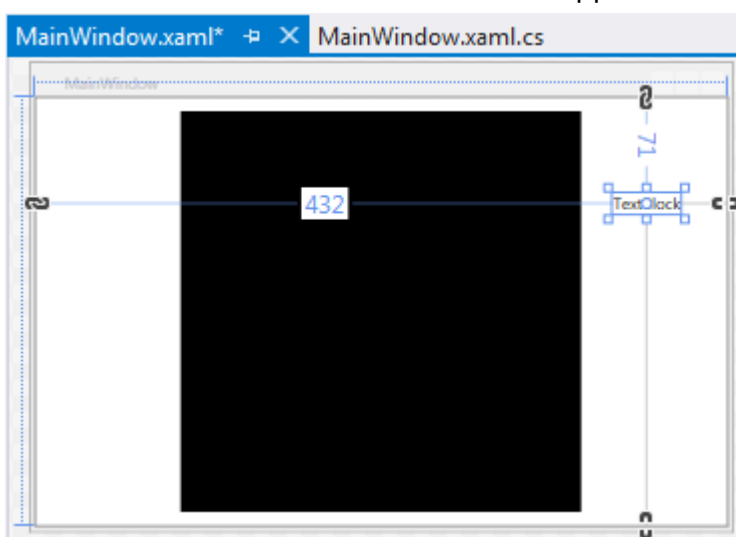
В раскрывающемся списке **Кисть** выбрать **Одноцветная кисть**.

В раскрывающемся списке **Макет** задать ширину (**Width**) и высоту (**Height**) по **300** пикселей, горизонтальное (**HorizontalAlignment**) и вертикальное (**VerticalAlignment**) выравнивание – **Stretch** (по ширине)

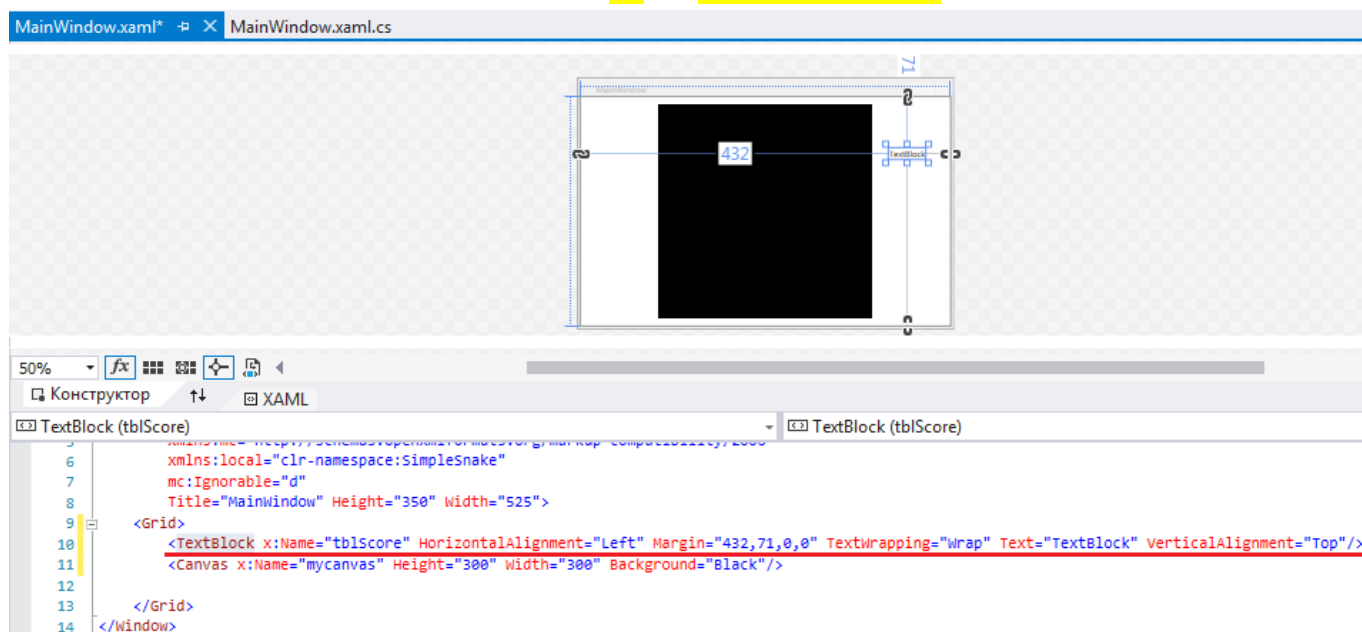


7. Одинарным кликом левой кнопкой мыши выделить в конструкторе окна элемент **TextBlock**.

В окне **Свойства** в поле **Имя** задать элементу **TextBlock** имя **tblScore**.



В окне разметки интерфейса в XAML найти строку описания элемента **TextBlock**, и заменить окончание строки с **/>** на **>0</TextBlock>**.



Например строка

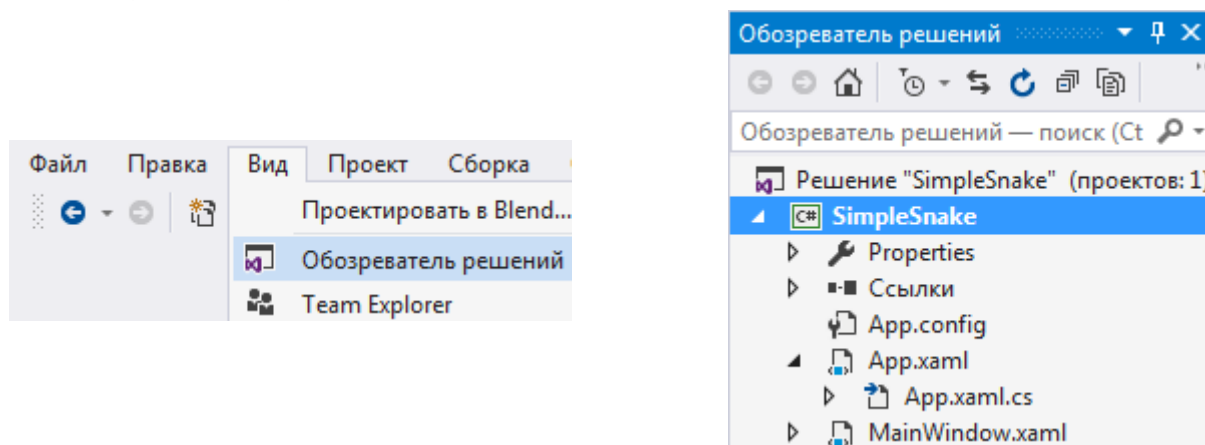
```
<TextBlock x:Name="textBlock" HorizontalAlignment="Left" Margin="432,71,0,0" TextWrapping="Wrap" Text="TextBlock" VerticalAlignment="Top" />
```

должна превратиться в

```
<TextBlock x:Name="textBlock" HorizontalAlignment="Left" Margin="432,71,0,0" TextWrapping="Wrap" Text="TextBlock" VerticalAlignment="Top">0</TextBlock>
```

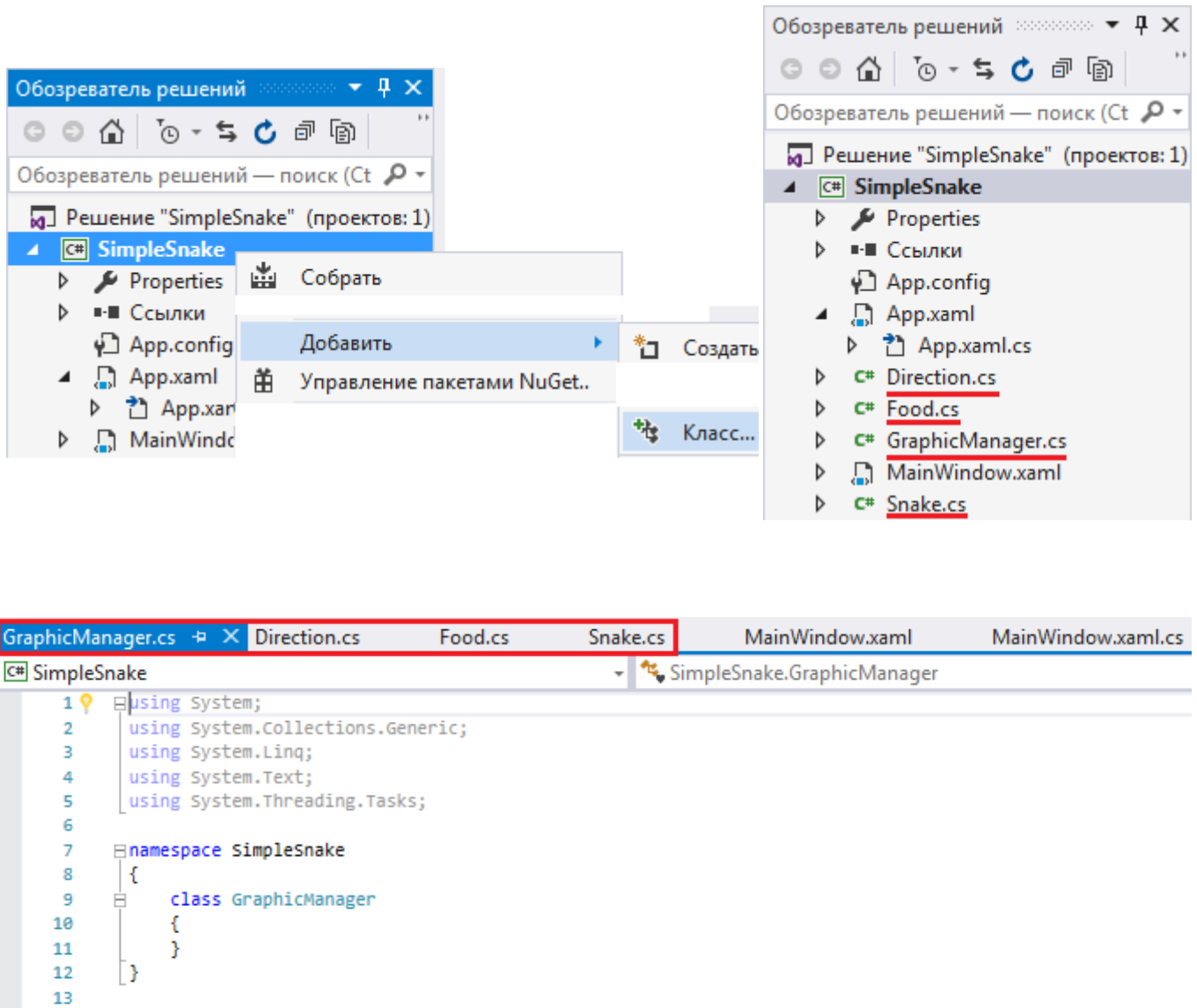
Данным изменением задаётся значение по-умолчанию элемента **TextBlock**, то есть какое значение будет отображаться в этом элементе при запуске программы.

8. Если не запущен **Обозреватель решений**, то запустить его через меню **Вид – Обозреватель решений**



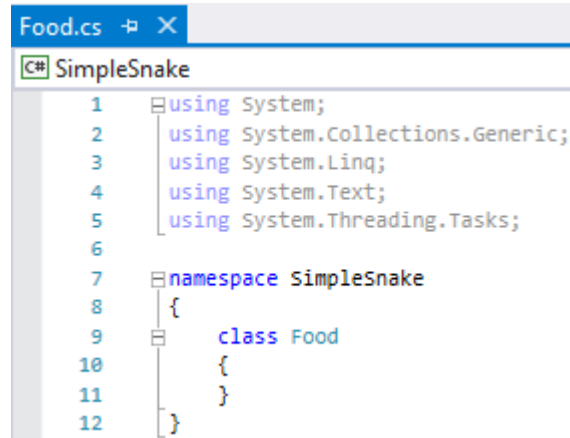
9. Добавить в проект **SimpleSnake** четыре новых класса с именами **Snake**, **Food**, **Direction**, **GraphicManager** (клик правой кнопкой мыши по проекту **SimpleSnake** – **Добавить** – **Класс** – задать имя класса – ОК)

Соответствующие классы появятся в **Обозревателе решений** и откроются соответствующие вкладки.



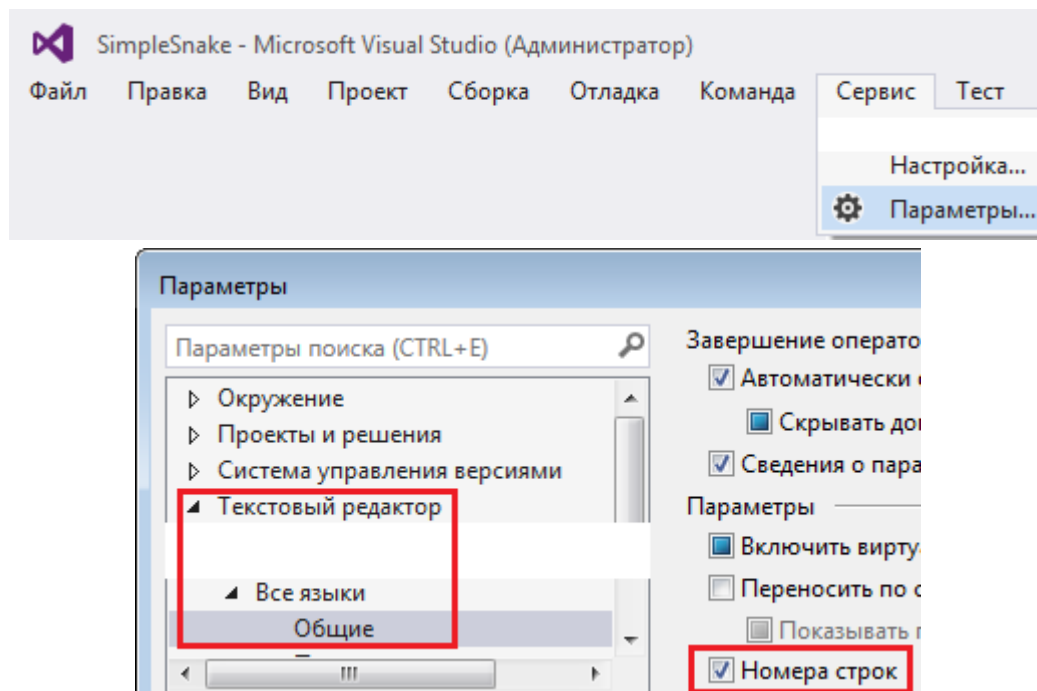


10. Перейти в класс **Food** (одинарный клик по соответствующей вкладке или двойной клик в **Обзревателе решений** по соответствующему классу)



```
Food.cs
C# SimpleSnake
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace SimpleSnake
8 {
9     class Food
10     {
11     }
12 }
```

Если необходимо включить нумерацию строк, то с панели меню перейти в **Сервис – Параметры – Текстовый редактор – Все языки – Общие** – установить галочку **Номера строк**



## 11. Изменить класс **Food** и раздел **using** следующим образом

```
Food.cs  +  X
C# SimpleSnake  SimpleSnake.Food

1  using System.Windows.Controls; // используется для объектов типа Canvas
2  using System.Windows.Media; // используется для объектов типа Brushes
3  using System.Windows.Shapes; // используется для объектов типа Ellipse
4  using System.Collections.Generic; // используется для объектов типа List
5
6  namespace SimpleSnake
7  {
8      public class Food
9      {
10         // объявление переменных класса (внутренних для данного класса)
11         public int CoordinateX, CoordinateY; // объявление переменных для координат
12         public readonly Ellipse Ellipse = new Ellipse(); // создание объекта эллипса-еды
13
14         // метод-конструктор класса Food - задание начальных значений
15         // переменным класса при создании объекта типа Food
16         // метод принимает на вход две целочисленные координаты
17         public Food(int _coordinateX, int _coordinateY)
18         {
19             CoordinateX = _coordinateX;
20             CoordinateY = _coordinateY;
21         }
22
23         // метод задания положения еды на поле
24         public void SetFoodPosition()
25         {
26             Ellipse.Width = Ellipse.Height = 10; // задание размеров в пикселях эллипса-еды
27             Ellipse.Fill = Brushes.Blue; // цвет заполнения эллипса-еды
28             Canvas.SetLeft(Ellipse, CoordinateX); // задание на поле левого верхнего положения
29             Canvas.SetTop(Ellipse, CoordinateY); // вершины прямоугольника, в который вписан эллипс
30         }
31
32         // метод добавления еды на поле
33         // метод принимает на вход два параметра: еду и поле для отображения
34         public void AddFoodInCanvas(List<Food> food, Canvas mycanvas)
35         {
36             food[0].SetFoodPosition(); // задание первому (и единственному) эллипсу-еды положения
37             mycanvas.Children.Insert(0, food[0].Ellipse); // вставка эллипса в указанную позицию
38         }
39     }
40 }
```

## 12. Перейти в класс Snake

```
Snake.cs -# X
C# SimpleSnake
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace SimpleSnake
8  {
9      class Snake
10     {
11     }
12 }
```

и изменить его и раздел **using** следующим образом

```
Snake.cs -# X
C# SimpleSnake
1  using System.Windows.Controls; // используется для объектов типа Canvas
2  using System.Windows.Media; // используется для объектов типа Brushes
3  using System.Windows.Shapes; // используется для объектов типа Ellipse
4  using System.Collections.Generic; // используется для объектов типа List
5
6  namespace SimpleSnake
7  {
8      public class Snake
9      {
10         // объявление переменных класса (внутренних для данного класса)
11         public int CoordinateX, CoordinateY; // объявление переменных для координат
12         public readonly Rectangle Rectangle = new Rectangle(); // создание объекта прямоугольника тела змеи
13
14         // метод-конструктор класса Food - задание начальных значений
15         // переменным класса при создании объекта типа Food
16         public Snake(int coordinateX, int coordinateY)
17         {
18             CoordinateX = coordinateX;
19             CoordinateY = coordinateY;
20         }
21
22         // метод задания положения змеи на поле
23         public void SetSnakePosition()
24         {
25             Rectangle.Width = Rectangle.Height = 10; // задание размеров в пикселях прямоугольника тела змеи
26             Rectangle.Fill = Brushes.Red; // цвет заполнения прямоугольника тела змеи
27             Canvas.SetLeft(Rectangle, CoordinateX); // задание на поле левого верхнего
28             Canvas.SetTop(Rectangle, CoordinateY); // положения вершины прямоугольника
29         }
30
31         // метод добавления прямоугольника-змеи на поле (змея изначально состоит из одного прямоугольника)
32         // метод принимает на вход два параметра: тело змеи и поле для отображения
33         public void AddSnakeInCanvas(List<Snake> snakebody, Canvas mycanvas)
34         {
35             foreach (Snake snake in snakebody) // для каждого элемента-прямоугольника в теле змеи
36             {
37                 snake.SetSnakePosition(); // задание элементу положения
38                 mycanvas.Children.Add(snake.Rectangle); // добавление элемента в коллекцию для отображения
39             }
40         }
41     }
42 }
```

```

42 // метод изменения положения змеи на каждом шаге
43 // метод принимает на вход два параметра: направление движения и тело змеи
44 public static void StepForward(int direction, List<Snake> snakebody)
45 {
46     if (direction != 0) // если змея начала движение в любом направлении
47     {
48         for (int i = snakebody.Count - 1; i > 0; i--) // для всех элементов тела змеи кроме головы
49             // начиная с последнего элемента (с хвоста змеи)
50             snakebody[i] = snakebody[i - 1]; // задать каждому элементу координаты предыдущего элемента
51         }
52     }
53 }
54
55 // метод проверки выхода за границы поля
56 // метод принимает на вход два параметра: тело змеи и окно программы
57 public static void ExitOutside(List<Snake> snakebody, MainWindow mnwnd)
58 {
59     if (snakebody[0].CoordinateX > 300 || snakebody[0].CoordinateY > 300 || // если голова змеи
60         snakebody[0].CoordinateX < 0 || snakebody[0].CoordinateY < 0) // выходит за границы поля
61     {
62         mnwnd.Close(); // закрыть окно программы
63     }
64 }
65
66 // метод проверки змеи на самосъедение
67 // метод принимает на вход два параметра: тело змеи и окно программы
68 public static void EatMyself(List<Snake> snakebody, MainWindow mnwnd)
69 {
70     for (int i = 1; i < snakebody.Count; i++) // для всех элементов тела змеи
71     {
72         if (snakebody[0].CoordinateX == snakebody[i].CoordinateX && // если координаты головы змеи и любого
73             snakebody[0].CoordinateY == snakebody[i].CoordinateY) // другого элемента тела змеи совпадают
74             mnwnd.Close(); // закрыть окно программы
75     }
76 }
77 }
78 }

```

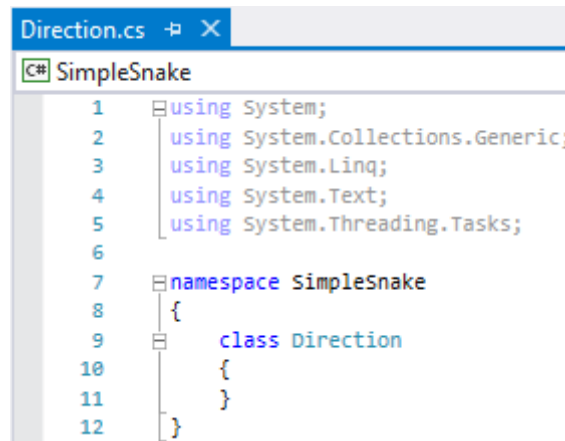
### 13. Перейти в класс **GraphicManager**

```
GraphicManager.cs  ▢ ×
C# SimpleSnake
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace SimpleSnake
8  {
9      class GraphicManager
10     {
11     }
12 }
```

и изменить его и раздел **using** следующим образом

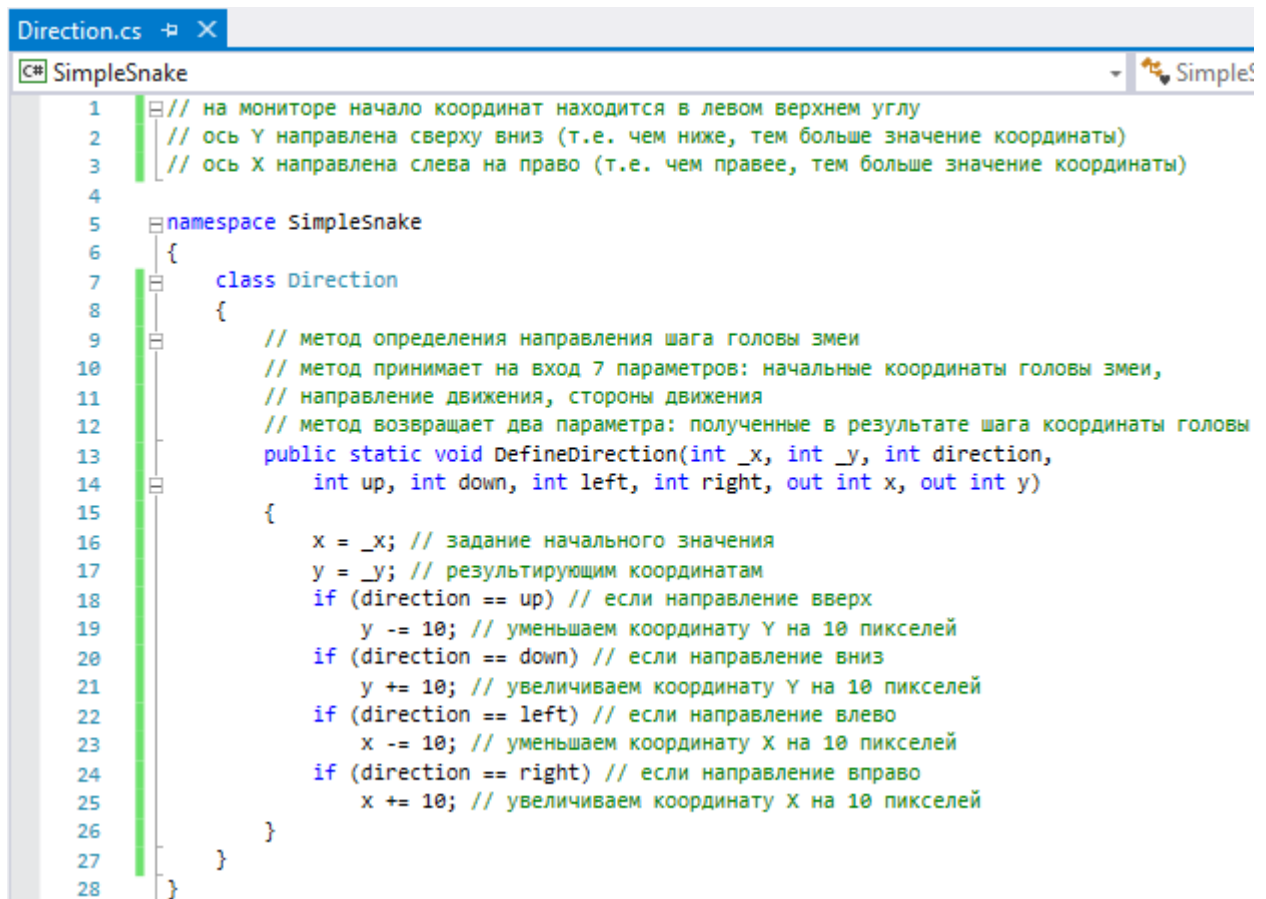
```
GraphicManager.cs  ▢ ×
C# SimpleSnake
using System; // используется для объектов типа Random
using System.Windows.Controls; // используется для объектов типа Canvas
using System.Windows.Shapes; // используется для объектов типа Ellipse
using System.Collections.Generic; // используется для объектов типа List
5
6 namespace SimpleSnake
7 {
8     public class GraphicManager
9     {
10         // объявление переменных класса (внутренних для данного класса)
11         Canvas mycanvas; // объявление переменной того же типа, что и поле игры
12         static int score; // объявление переменной для счёта съеденной еды
13
14         // метод обработки съедания змеей еды
15         // метод принимает на вход 5 параметров: тело змеи, еду, поле игры, текстовый блок счёта, генератор случайных чисел
16         public static void EatFood(List<Snake> snakebody, List<Food> food, Canvas mycanvas, TextBlock tblScore, Random rnd)
17         {
18             // если координаты головы змеи совпадают с координатами еды на поле
19             if (snakebody[0].CoordinateX == food[0].CoordinateX && snakebody[0].CoordinateY == food[0].CoordinateY)
20             {
21                 snakebody.Add(new Snake(food[0].CoordinateX, food[0].CoordinateY)); // добавить к телу змеи новый элемент
22                 food[0] = new Food(rnd.Next(0, 29) * 10, rnd.Next(0, 29) * 10); // создать новый элемент еды в новой позиции
23                 mycanvas.Children.RemoveAt(0); // удаление съеденной еды с поля
24                 food[0].AddFoodInCanvas(food, mycanvas); // добавление новой еды на поле
25                 score++; // увеличение счёта на единицу
26                 tblScore.Text = score.ToString(); // отображение счёта в текстовом блоке в окне программы
27             }
28         }
29
30         // метод очистки игрового поля
31         // метод принимает на вход два параметра: тело змеи и поле игры
32         public static void ClearWindow(List<Snake> snakebody, Canvas mycanvas)
33         {
34             int count = 0; // объявление и задание значения внутренней переменной метода
35
36             for (int i = 0; i < mycanvas.Children.Count; i++) // для каждого элемента на игровом поле
37             {
38                 if (mycanvas.Children[i] is Rectangle) // если элемент поля прямоугольник (т.е. элемент змеи)
39                     count++; // увеличить счётчик на единицу
40             }
41             mycanvas.Children.RemoveRange(1, count); // удалить все элементы змеи с поля
42             count = 0; // обнулить счётчик элементов тела змеи
43             snakebody[0].AddSnakeInCanvas(snakebody, mycanvas); // добавление головы змеи на поле
44         }
45     }
46 }
```

## 14. Перейти в класс **Direction**



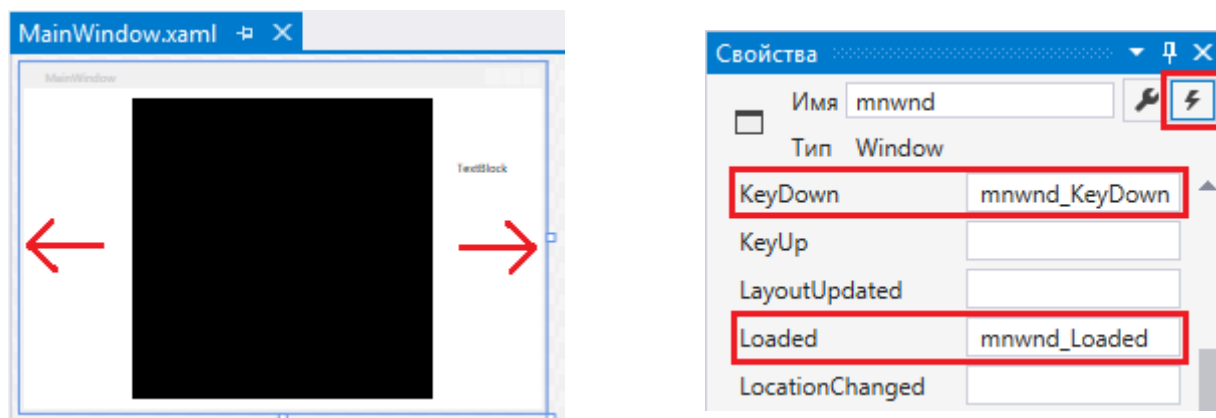
```
Direction.cs
C# SimpleSnake
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace SimpleSnake
8 {
9     class Direction
10    {
11    }
12 }
```

и изменить его и раздел **using** следующим образом

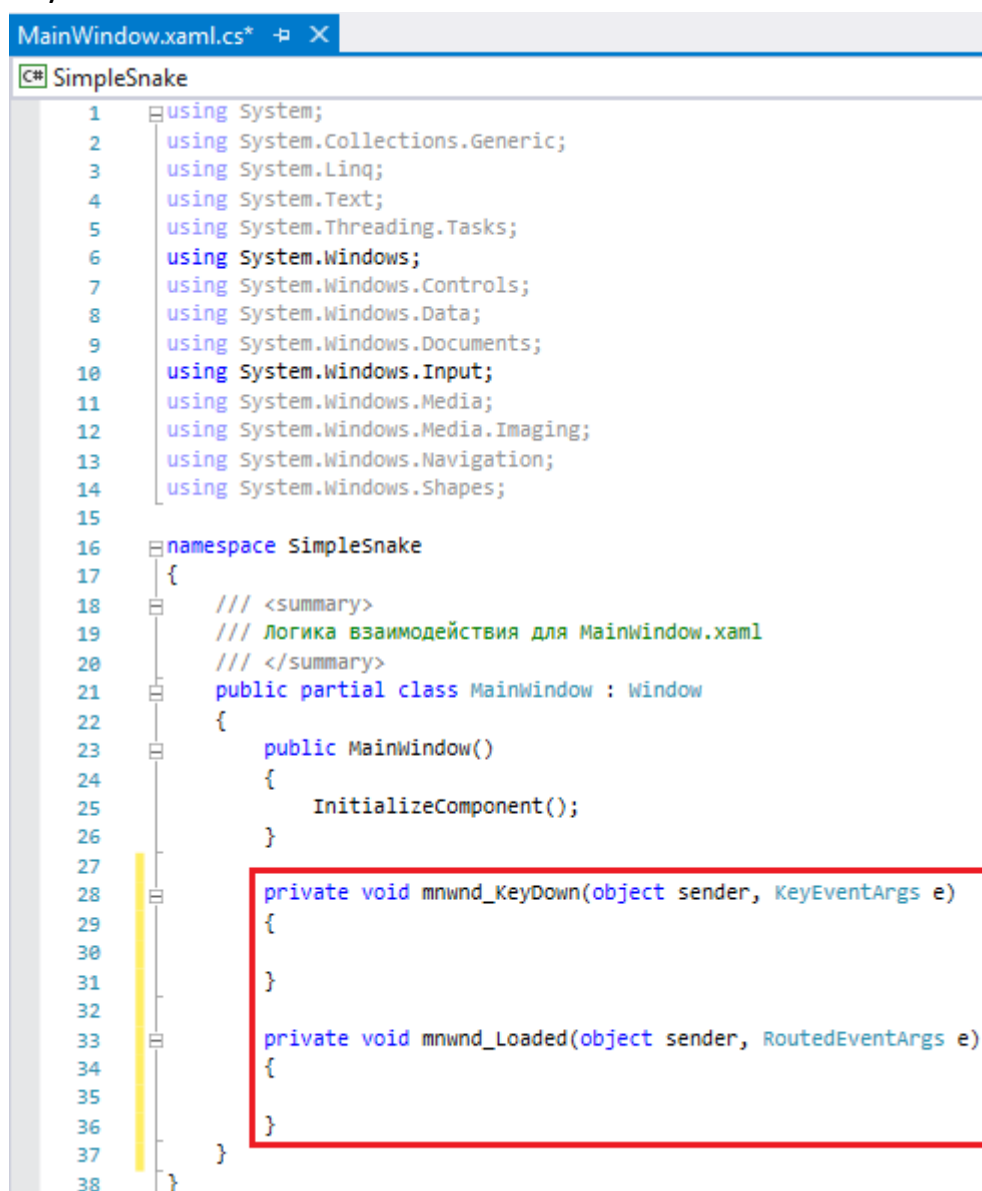


```
Direction.cs
C# SimpleSnake
1 // на мониторе начало координат находится в левом верхнем углу
2 // ось Y направлена сверху вниз (т.е. чем ниже, тем больше значение координаты)
3 // ось X направлена слева на право (т.е. чем правее, тем больше значение координаты)
4
5 namespace SimpleSnake
6 {
7     class Direction
8     {
9         // метод определения направления шага головы змеи
10        // метод принимает на вход 7 параметров: начальные координаты головы змеи,
11        // направление движения, стороны движения
12        // метод возвращает два параметра: полученные в результате шага координаты головы
13        public static void DefineDirection(int _x, int _y, int direction,
14            int up, int down, int left, int right, out int x, out int y)
15        {
16            x = _x; // задание начального значения
17            y = _y; // результирующим координатам
18            if (direction == up) // если направление вверх
19                y -= 10; // уменьшаем координату Y на 10 пикселей
20            if (direction == down) // если направление вниз
21                y += 10; // увеличиваем координату Y на 10 пикселей
22            if (direction == left) // если направление влево
23                x -= 10; // уменьшаем координату X на 10 пикселей
24            if (direction == right) // если направление вправо
25                x += 10; // увеличиваем координату X на 10 пикселей
26        }
27    }
28 }
```

15. Перейти в конструктор окна приложения (**MainWindow.xaml**), одинарным кликом левой кнопки мыши выделить окно **MainWindow**,



в окне **Свойства** перейти на вкладку **Обработчик событий**, найти события **KeyDown** и **Loaded**, и сделать двойной клик по пустому полю справа от каждого события. В результате для каждого из событий автоматически сгенерируются методы **mnwnd\_KeyDown** и **mnwnd\_Loaded**. Эти методы автоматически пропишутся в классе **MainWindow.xaml.cs**.



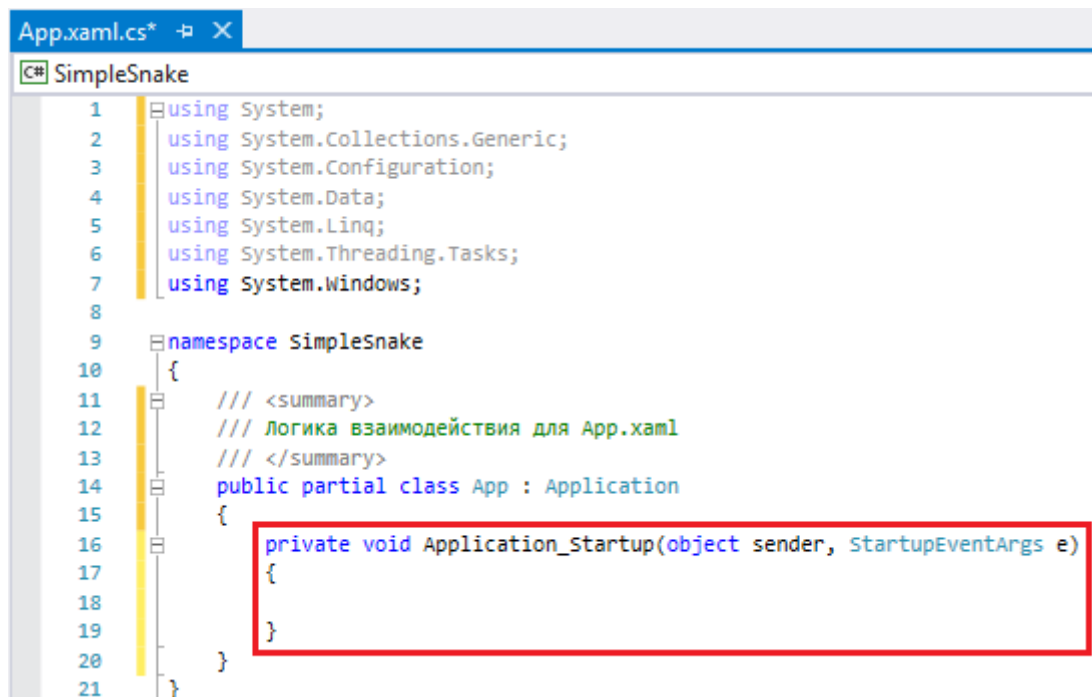
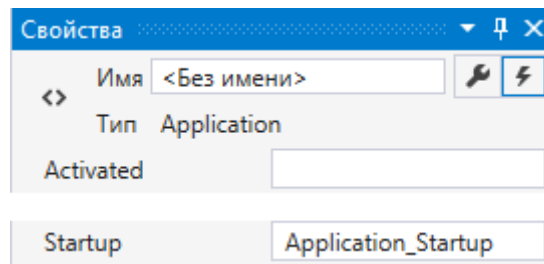
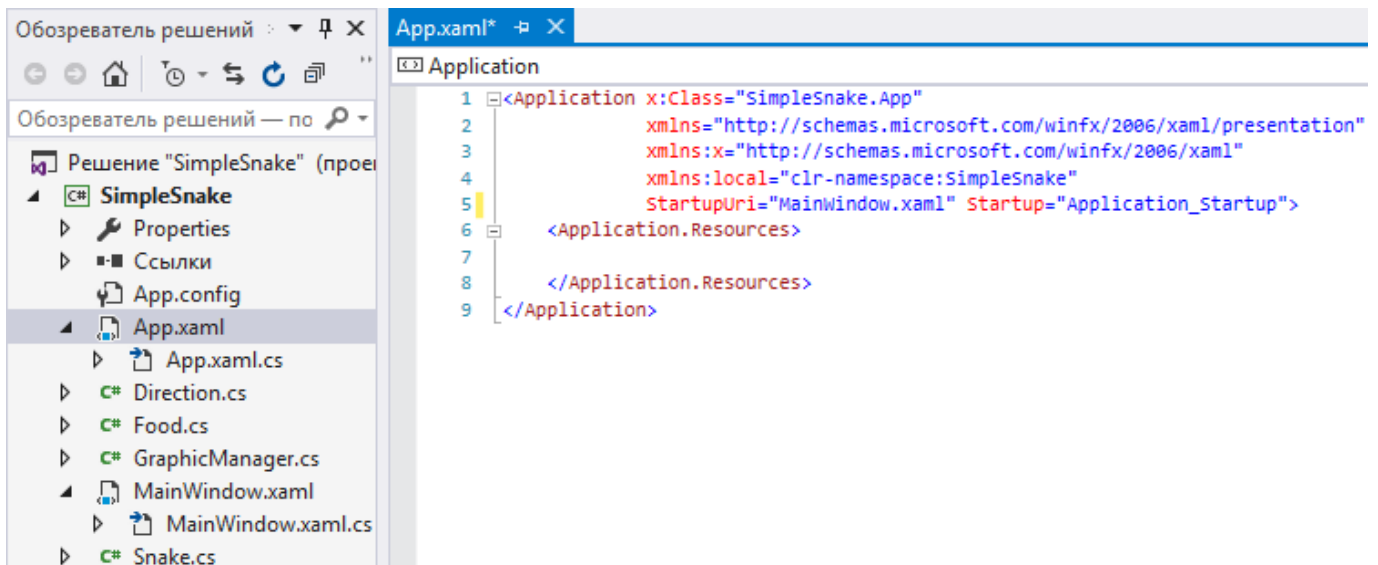


## 16. В классе **MainWindow.xaml.cs** изменить его и раздел **using** следующим образом

```
MainWindow.xaml.cs - X
C# SimpleSnake SimpleSnake.MainWindow
1 using System; // используется для объектов типа Random
2 using System.Windows; // используется для объектов типа RoutedEventArgs
3 using System.Windows.Input; // используется для объектов типа Key и KeyEventArgs
4 using System.Collections.Generic; // используется для объектов типа List
5 using System.Windows.Threading; // используется для объектов типа DispatcherTimer
6
7 namespace SimpleSnake
8 {
9     public partial class MainWindow : Window
10    {
11        // объявление переменных класса (внутренних для данного класса)
12        DispatcherTimer time; // объявление таймера
13        List<Snake> snakebody; // объявление тела змеи
14        List<Food> food; // объявление еды
15        GraphicManager graphManager; // объявление переменной для отображения
16        Random rnd = new Random(); // создание генератора случайных чисел
17        int x = 100; // объявление и задание начального значения голове змеи
18        int y = 100; // объявление и задание начального значения голове змеи
19        int direction; // объявление направления движения голове змеи
20        int left = 4; // объявление и задание значения положению "влево"
21        int right = 6; // объявление и задание значения положению "вправо"
22        int up = 8; // объявление и задание значения положению "вверх"
23        int down = 2; // объявление и задание значения положению "вниз"
24
25        public MainWindow()
26        {
27            InitializeComponent();
28            time = new DispatcherTimer(); // присвоение таймеру нового пустого объекта
29            snakebody = new List<Snake>(); // присвоение телу-голове змеи нового пустого объекта
30            food = new List<Food>(); // присвоение еде нового пустого объекта
31            snakebody.Add(new Snake(x, y)); // задание голове змеи начальных координат
32            food.Add(new Food(rnd.Next(0, 29) * 10, rnd.Next(0, 29) * 10)); // задание еде начальных координат
33            time.Interval = new TimeSpan(0, 0, 0, 0, 100); // задание интервала обновления таймера в миллисекундах
34            time.Tick += timer_TickTick; // через каждый интервал таймера выполнять метод timer_TickTick
35        }
36
37        // метод выполняющийся после каждого интервала таймера
38        void timer_TickTick(object sender, EventArgs e)
39        {
40            Snake.StepForward(direction, snakebody); // тело змеи делает один шаг в указанном направлении
41            Direction.DefineDirection(x, y, direction, up, down, left, right, out x, out y); // голова змеи делает шаг
42            GraphicManager.EatFood(snakebody, food, myCanvas, tblScore, rnd); // проверка съедения еды
43            snakebody[0] = new Snake(x, y); // изменение положения головы змеи
44            Snake.ExitOutside(snakebody, mwWnd); // проверка выхода змеи за границы поля
45            Snake.EatMyself(snakebody, mwWnd); // проверка змеи на самосъедение
46            GraphicManager.ClearWindow(snakebody, myCanvas); // очистка поля
47        }
48
49        // метод определения нажатой клавиши
50        // в методе реализован запрет движения змеи назад, только прямо или вбок от текущего направления
51        private void mwWnd_KeyDown(object sender, KeyEventArgs e)
52        {
53            if (e.Key == Key.Up && direction != down) // если нажата стрелка вверх и текущее направление не вниз
54                direction = up; // тогда задаём направление движения вверх
55            if (e.Key == Key.Down && direction != up)
56                direction = down;
57            if (e.Key == Key.Left && direction != right)
58                direction = left;
59            if (e.Key == Key.Right && direction != left)
60                direction = right;
61        }
62
63        // метод выполняется при загрузке окна программы
64        private void mwWnd_Loaded(object sender, RoutedEventArgs e)
65        {
66            snakebody[0].AddSnakeInCanvas(snakebody, myCanvas); // добавить змею на поле
67            food[0].AddFoodInCanvas(food, myCanvas); // добавить еду на поле
68            time.Start(); // запустить таймер
69        }
70    }
71 }
```



17. Перейти в класс **App.xaml** – окно **Свойства** – вкладка **Обработчики событий** – двойной клик по полю справа от события **Startup** – автоматически сгенерируется метод **Application\_Startup** в классе **App.xaml.cs**



18. Добавить в метод **Application\_Startup** следующий код

```
public partial class App : Application
{
    // метод выполняющийся при запуске программы
    private void Application_Startup(object sender, StartupEventArgs e)
    {
        MainWindow = new MainWindow(); // создать окно программы
    }
}
```

19. Запустить программу

