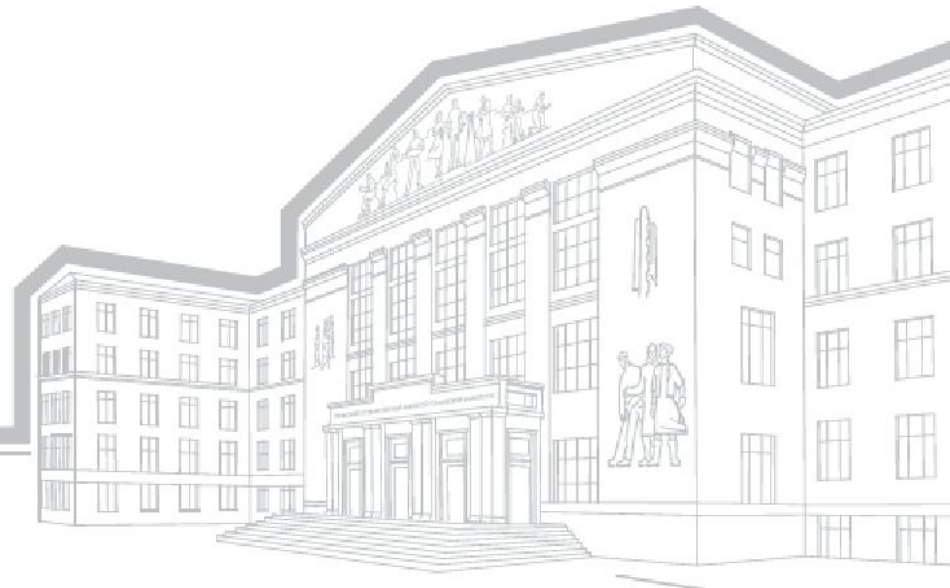
УФИМСКИЙ ГОСУДАРСТВЕННЫЙ
НЕФТЯНОЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Визуальное программирование

Чиганова Наталья Викторовна
к.ф.-м.н., доцент кафедры
«Цифровые технологии и
моделирование»



Лекция 2. Компоновка

Компоновка (layout) представляет собой процесс размещения элементов внутри контейнера.

Благодаря компоновке возможно настроить элементы интерфейса, позиционировать их определенным образом.

Например, элементы компоновки в WPF позволяют при ресайзе - сжатии или растяжении масштабировать элементы, что очень удобно, а визуально не создает всяких шероховатостей типа незаполненных пустот на форме

В WPF компоновка осуществляется при помощи специальных контейнеров. Фреймворк предоставляет нам следующие контейнеры:

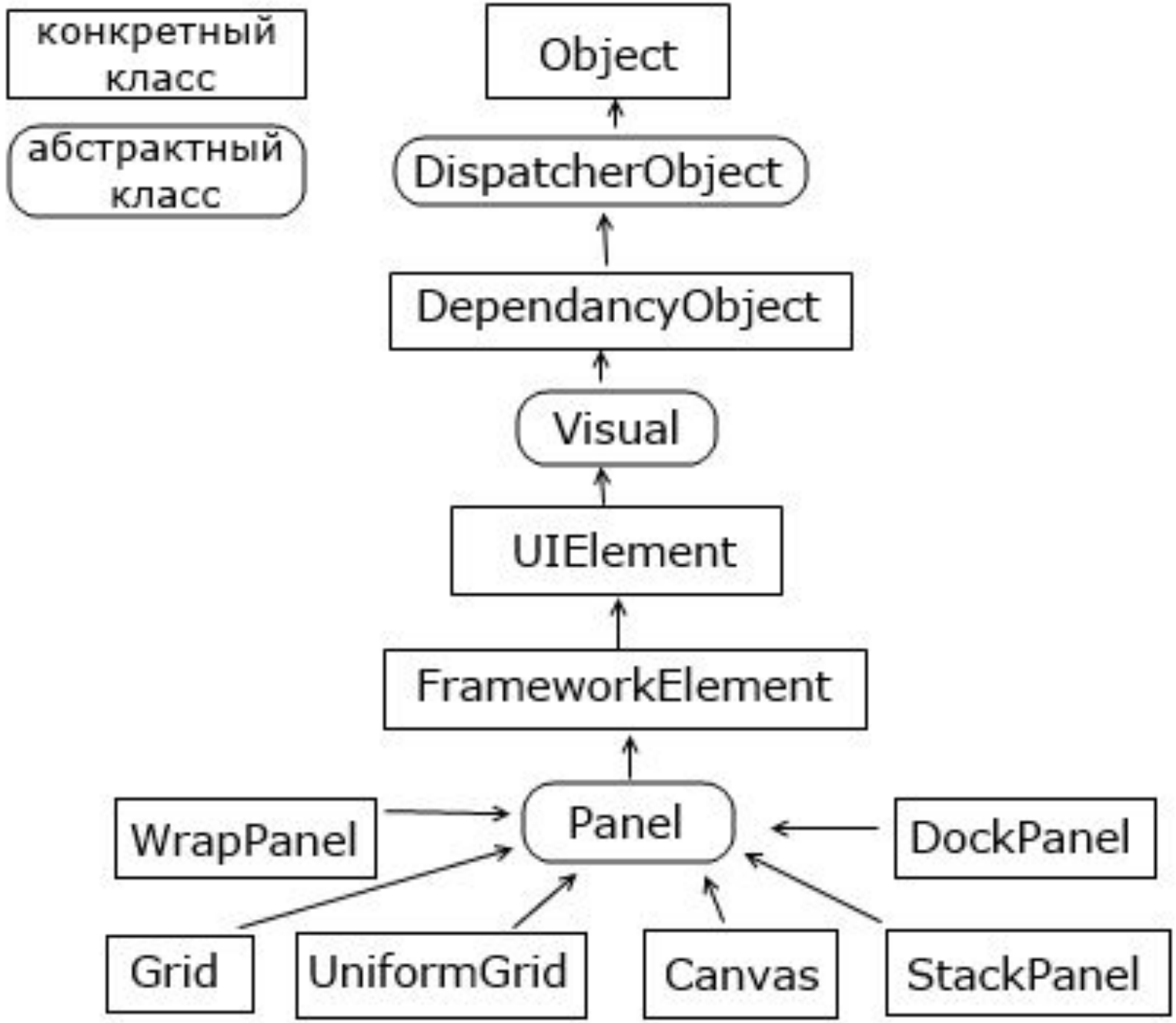
- Grid,
- UniformGrid,
- StackPanel,
- WrapPanel,
- DockPanel
- Canvas.

Различные контейнеры могут содержать внутри себя другие контейнеры.

Кроме данных контейнеров существует еще ряд элементов, такие как TabPanel, которые могут включать другие элементы и даже контейнеры компоновки, однако на саму компоновку не столь влияют в отличие от выше перечисленных. Кроме того, если нам не хватает стандартных контейнеров, мы можем определить свои с нужной нам функциональностью.

Контейнеры компоновки позволяют эффективно распределить доступное пространство между элементами, найти для него наиболее предпочтительные размеры.

Все выше перечисленные контейнеры компоновки наследуются от абстрактного класса Panel.


конкретный класс
абстрактный класс
Object
DispatcherObject
DependancyObject
Visual
UIElement
FrameworkElement
Panel
WrapPanel
DockPanel
Grid
UniformGrid
Canvas
StackPanel

Процесс компоновки

Процесс компоновки проходит два этапа:

- измерение (measure)
- расстановка (arrange).

На этапе измерения контейнер производит измерение предпочтительного для дочерних элементов места. Однако не всегда контейнер имеет достаточно места, чтобы расставить все элементы по их предпочтительным размером, поэтому их размеры приходится усекать.

Затем происходит этап непосредственной расстановки дочерних элементов внутри контейнера.

Grid

Это наиболее мощный и часто используемый контейнер, напоминающий обычную таблицу. Он содержит столбцы и строки, количество которых задает разработчик.

Для определения строк используется свойство **RowDefinitions**, а для определения столбцов - свойство **ColumnDefinitions**

```
<Grid.RowDefinitions>
    <RowDefinition></RowDefinition>
    <RowDefinition></RowDefinition>
    <RowDefinition></RowDefinition>
</Grid.RowDefinitions>
<Grid.ColumnDefinitions>
    <ColumnDefinition></ColumnDefinition>
    <ColumnDefinition></ColumnDefinition>
    <ColumnDefinition></ColumnDefinition>
</Grid.ColumnDefinitions>
```

Каждая строка задается с помощью вложенного элемента RowDefinition, который имеет открывающий и закрывающий тег. При этом задавать дополнительную информацию необязательно. То есть в данном случае у нас определено в гриде 3 строки.

Каждая столбец задается с помощью вложенного элемента ColumnDefinition.

таблица 3x3.

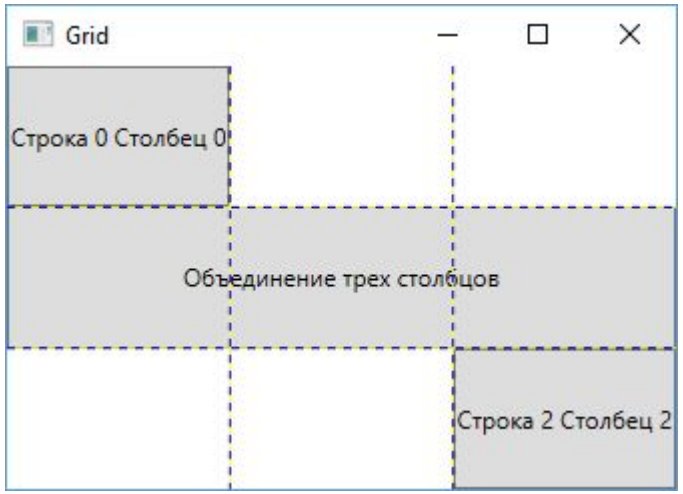
Чтобы задать позицию элемента управления с привязкой к определенной ячейке Grid, в разметке элемента нужно прописать значения свойств Grid.Column и Grid.Row, тем самым указывая, в каком столбце и строке будет находиться элемент.

Кроме того, если мы хотим растянуть элемент управления на несколько строк или столбцов, то можно указать свойства Grid.ColumnSpan и Grid.RowSpan.

Пример

```
<Window x:Class="LayoutApp.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:LayoutApp"
        mc:Ignorable="d"
        Title="Grid" Height="250" Width="350">
    <Grid ShowGridLines="True">
        <Grid.RowDefinitions>
            <RowDefinition></RowDefinition>
            <RowDefinition></RowDefinition>
            <RowDefinition></RowDefinition>
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition></ColumnDefinition>
            <ColumnDefinition></ColumnDefinition>
            <ColumnDefinition></ColumnDefinition>
        </Grid.ColumnDefinitions>
        <Button Grid.Column="0" Grid.Row="0" Content="Строка 0 Столбец 0"  />
        <Button Grid.Column="0" Grid.Row="1" Content="Объединение трех столбцов" Grid.ColumnSpan="3"
/>
        <Button Grid.Column="2" Grid.Row="2" Content="Строка 2 Столбец 2"  />
    </Grid>
</Window>
```

Атрибут ShowGridLines="True" у элемента Grid задает видимость сетки, по умолчанию оно равно False.



Установка размеров

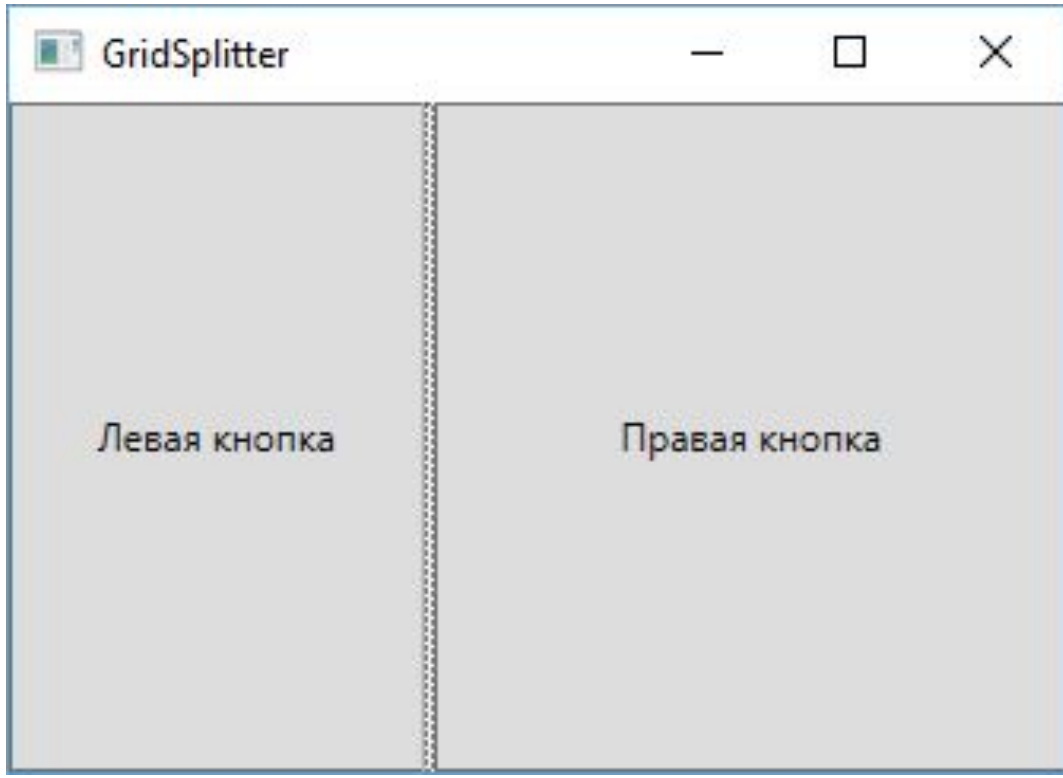
Автоматические размеры	<ColumnDefinition Width="Auto" /> <RowDefinition Height="Auto" />
Абсолютные размеры	ColumnDefinition Width="150" /> <RowDefinition Height="150" />
Пропорциональные размеры	<ColumnDefinition Width="*" /> <ColumnDefinition Width="0.5*" /> <ColumnDefinition Width="1.5*" />

UniformGrid

```
<UniformGrid Rows="2" Columns="2">
    <Button Content="Left Top" />
    <Button Content="Right Top" />
    <Button Content="Left Bottom" />
    <Button Content="Right Bottom" />
</UniformGrid>
```

GridSplitter

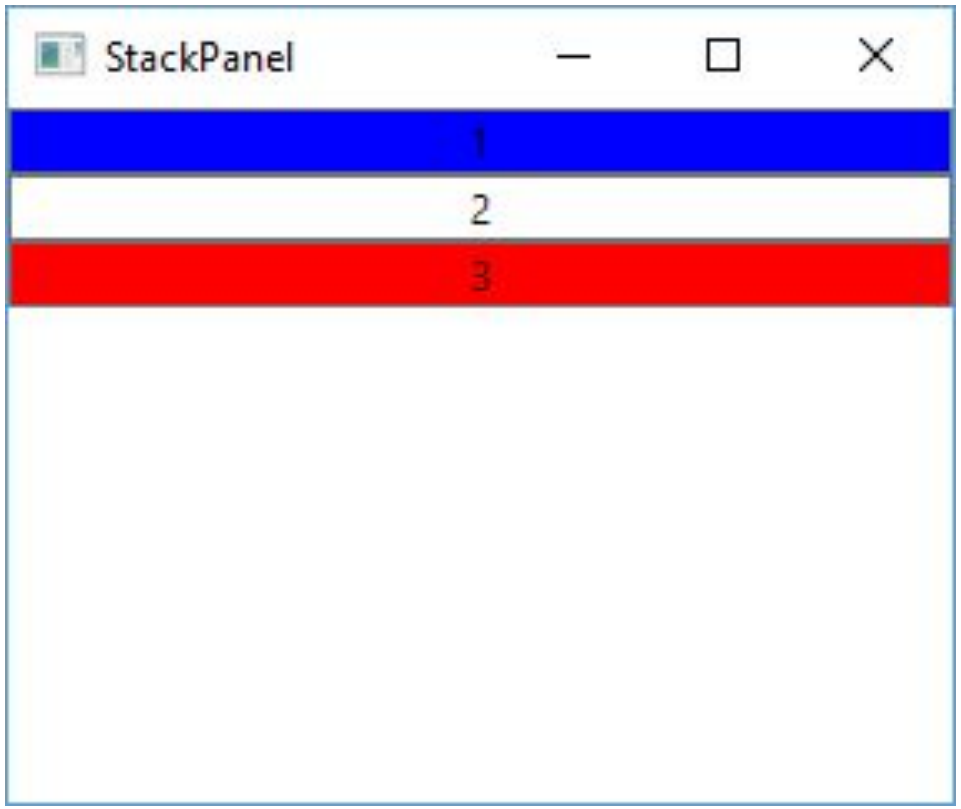
```
<Grid>
   <Grid.ColumnDefinitions>
      <ColumnDefinition Width="*" />
      <ColumnDefinition Width="Auto" />
      <ColumnDefinition Width="*" />
   </Grid.ColumnDefinitions>
   <Button Grid.Column="0" Content="Левая кнопка"
/>
   <GridSplitter Grid.Column="1"
ShowsPreview="False" Width="3"
      HorizontalAlignment="Center"
VerticalAlignment="Stretch" />
   <Button Grid.Column="2" Content="Правая
кнопка" />
</Grid>
```



StackPanel

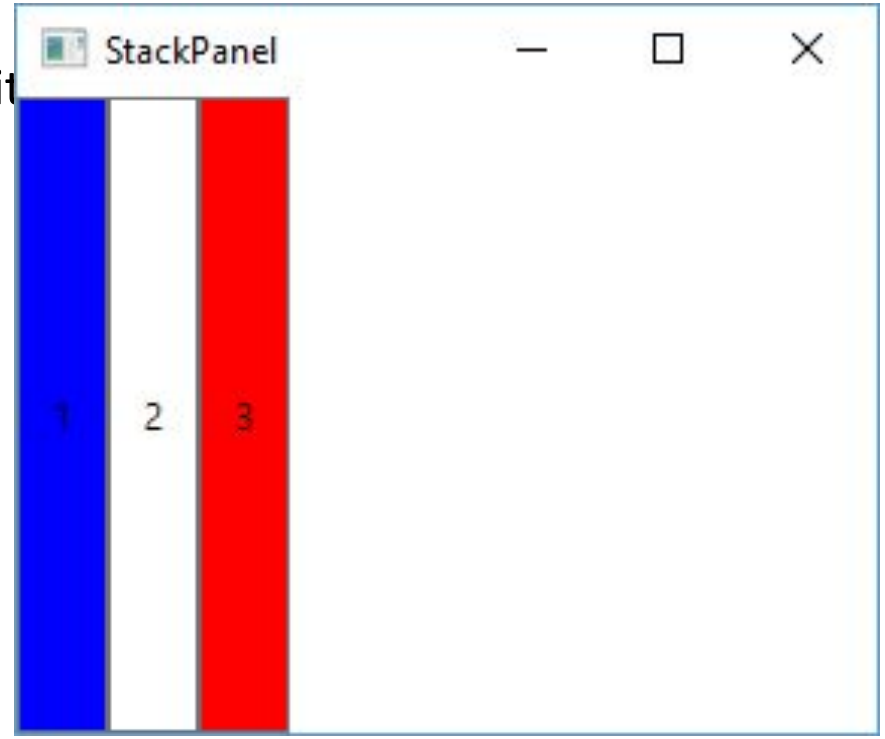
Это более простой элемент компоновки. Он располагает все элементы в ряд либо по горизонтали, либо по вертикали в зависимости от ориентации

```
Window x:Class="LayoutApp.MainWindow"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:local="clr-namespace:LayoutApp"
      mc:Ignorable="d"
      Title="StackPanel" Height="300" Width="300">
   <Grid>
      <StackPanel>
         <Button Background="Blue" Content="1" />
         <Button Background="White" Content="2" />
         <Button Background="Red" Content="3" />
      </StackPanel>
   </Grid>
</Window>
```



```
<Window x:Class="LayoutApp.MainWindow"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibilit
6"
      xmlns:local="clr-namespace:LayoutApp"
      mc:Ignorable="d"
      Title="StackPanel" Height="300" Width="300">
   <StackPanel Orientation="Horizontal">
      <Button Background="Blue" MinWidth="30" Content="1" />
      <Button Background="White" MinWidth="30" Content="2" />
      <Button Background="Red" MinWidth="30" Content="3" />
   </StackPanel>
</Window>
```



При горизонтальной ориентации все вложенные элементы располагаются слева направо.

Если мы хотим, чтобы наполнение стека начиналось справа налево, то нам надо задать свойство **FlowDirection**:

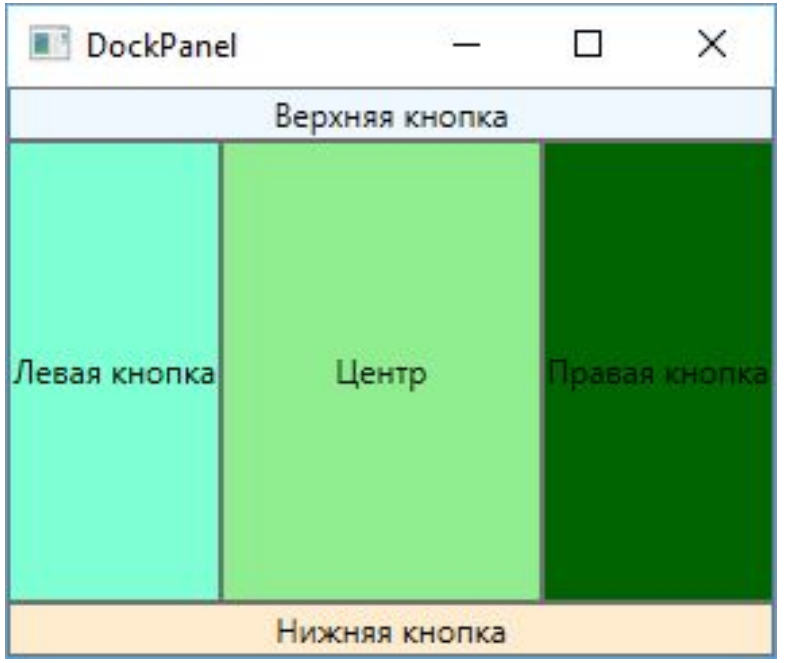
<StackPanel Orientation="Horizontal" FlowDirection="RightToLeft">.

По умолчанию это свойство имеет значение LeftToRight - то есть слева направо.

DockPanel

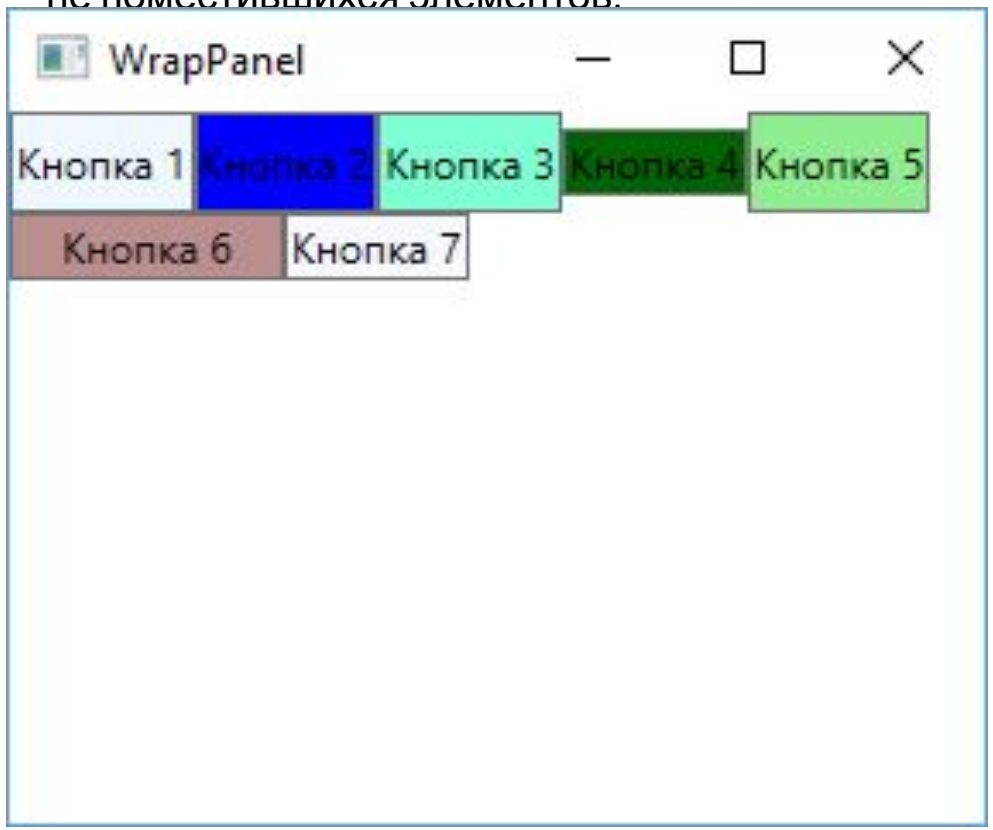
Этот контейнер прижимает свое содержимое к определенной стороне внешнего контейнера. Для этого у вложенных элементов надо установить сторону, к которой они будут прижиматься с помощью свойства DockPanel.Dock.

```
<Window x:Class="LayoutApp.MainWindow"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:local="clr-namespace:LayoutApp"
      mc:Ignorable="d"
      Title="DockPanel" Height="250" Width="300">
   <DockPanel LastChildFill="True">
      <Button DockPanel.Dock="Top" Background="AliceBlue" Content="Верхняя кнопка" />
      <Button DockPanel.Dock="Bottom" Background="BlanchedAlmond" Content="Нижняя кнопка" />
      <Button DockPanel.Dock="Left" Background="Aquamarine" Content="Левая кнопка" />
      <Button DockPanel.Dock="Right" Background="DarkGreen" Content="Правая кнопка" />
      <Button Background="LightGreen" Content="Центр" />
</DockPanel>
</Window>
```



WrapPanel

Эта панель, подобно StackPanel. Главное отличие от StackPanel - если элементы не помещаются в строке или столбце, создаются новые столбец или строка для не поместившихся элементов.

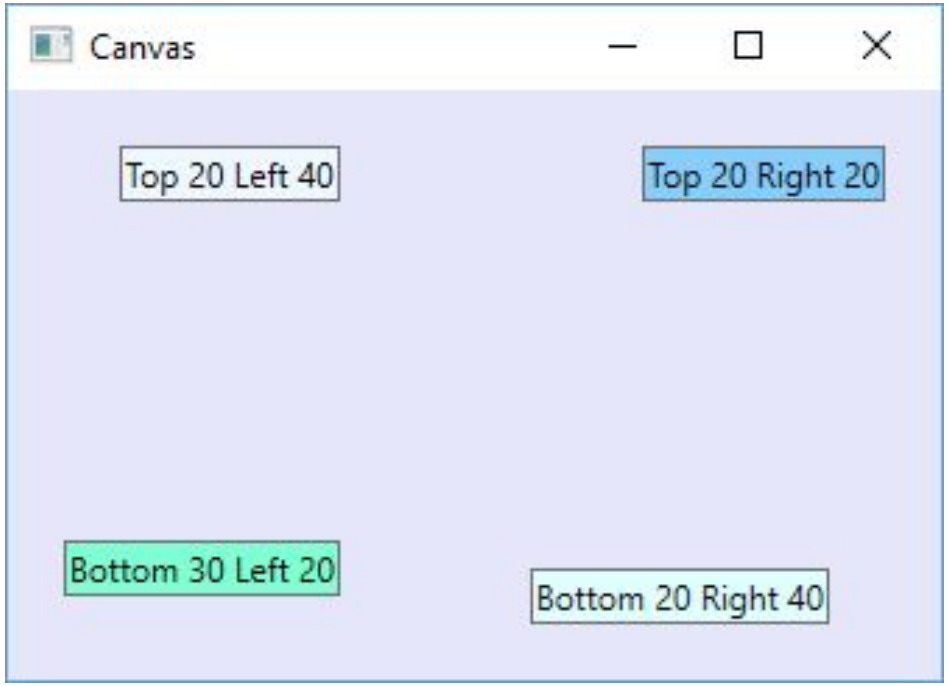



```
<Window x:Class="LayoutApp.MainWindow"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
      xmlns:local="clr-namespace:LayoutApp"
      mc:Ignorable="d"
      Title="WrapPanel" Height="250" Width="300">
   <WrapPanel>
      <Button Background="AliceBlue" Content="Кнопка 1" />
      <Button Background="Blue" Content="Кнопка 2" />
      <Button Background="Aquamarine" Content="Кнопка 3"
Height="30"/>
      <Button Background="DarkGreen" Content="Кнопка 4"
Height="20"/>
      <Button Background="LightGreen" Content="Кнопка 5"/>
      <Button Background="RosyBrown" Content="Кнопка 6"
Width="80" />
      <Button Background="GhostWhite" Content="Кнопка 7" />
   </WrapPanel>
</Window>
```

Canvas




```
<Window x:Class="Layout.MainWindow"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
      Title="MainWindow" Height="300" Width="300">
   <Grid>
      <Canvas Background="Lavender">
         <Button Background="AliceBlue" Content="Top 20 Left 40"
Canvas.Top="20" Canvas.Left="40" />
         <Button Background="LightSkyBlue" Content="Top 20 Right 20"
Canvas.Top="20" Canvas.Right="20"/>
         <Button Background="Aquamarine" Content="Bottom 30 Left
20" Canvas.Bottom="30" Canvas.Left="20"/>
         <Button Background="LightCyan" Content="Bottom 20 Right
40" Canvas.Bottom="20" Canvas.Right="40"/>
      </Canvas>
   </Grid>
</Window>
```

Свойства компоновки элементов. *Выравнивание*

Свойство **HorizontalAlignment**

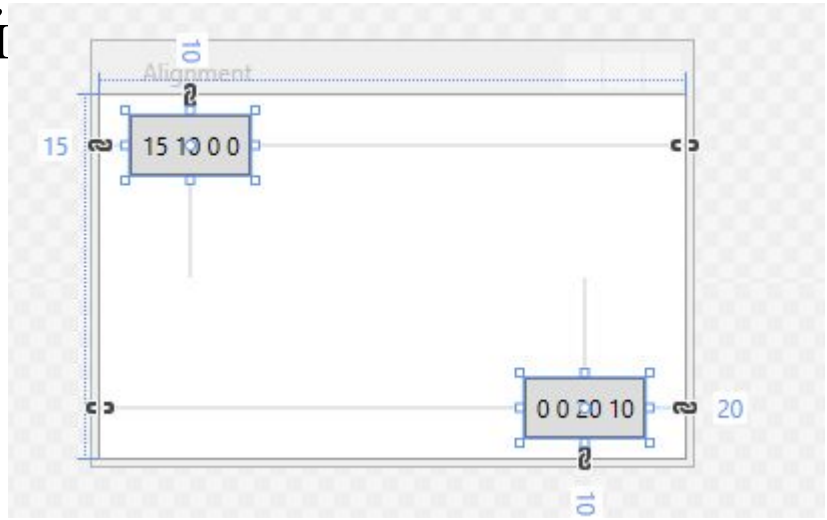
выравнивает элемент по горизонтали относительно правой или левой стороны контейнера и соответственно может принимать значения **Left**, **Right**, **Center** (положение по центру), **Stretch** (растяжение по всей ширине).

Свойство **VerticalAlignment**

выравнивание по вертикали с помощью свойства **VerticalAlignment**, которое принимает следующие значения: **Top** (положение в верху контейнера), **Bottom** (положение внизу), **Center** (положение по центру), **Stretch** (растяжение по всей высоте).

Отступы margin

Свойство **Margin** устанавливает отступы вокруг элемента. Синтаксис: Margin=" левый_отступ верхний_отступ правый_отступ нижний_отступ". Например, установим отступы у одной кнопки слева и сверху, а у другой




```
<Grid>
    <Button Content="15 10 0 0"
Width="60" Height="30" Margin
="15 10 0 0"

HorizontalAlignment="Left"
VerticalAlignment="Top"/>
    <Button Content="0 0 20 10"
Width="60" Height="30" Margin ="0
0 20 10"
HorizontalAlignment="Right"
VerticalAlignment="Bottom"/>
</Grid>
```

Panel.ZIndex

```
<Grid>
    <Button Width="60" Height="30"
Panel.ZIndex="2" Margin="10 10 0
0">Один</Button>
    <Button Width="60" Height="30"
Panel.ZIndex="1" Margin="45 45 0
0">Два</Button>
    <Button Width="60" Height="30"
Panel.ZIndex="0" Margin="75 75 0
0">Три</Button>
</Grid>
```