Занятие 1

Основные понятия

Список источников

- Сайт: https://docs.microsoft.com/ru-ru/ и https://metanit.com/
- Exam Ref 70-483 Programming in C#, Wouter de Kort, O´Reilly
- CLR via C#. Программирование на платформе Microsoft.NET Framework 4.5 на языке C#. Рихтер
- Макдональд, WPF: Windows Presentation Foundation в .NET 4.5 с примерами на C# 5.0 для профессионалов
- Freeman, Pro ASP.NET MVC 5
- HTTP: The Definitive Guide by Brian Totty, David

Понимание делегатов

```
public delegate int Calculate(int x, int y);

  public int Add(int x, int y) { return x + y; }
  public int Multiply(int x, int y) { return x * y; }

  public void UseDelegate()
  {
    Calculate calc = Add;
    Console.WriteLine(calc(3, 4));

    calc = Multiply;
    Console.WriteLine(calc(3, 4));
  }
```

multicast delegate

```
public void MethodOne()
{
  Console.WriteLine("MethodOne");
}
public void MethodTwo()
{
  Console.WriteLine("MethodTwo");
}
public delegate void Del();

public void Multicast()
{
  Del d = MethodOne;
  d += MethodTwo;
  d();
}
```

Covariance with delegates

```
public delegate TextWriter CovarianceDel();

public StreamWriter MethodStream() { return null; }
public StringWriter MethodString() { return null; }

CovarianceDel del;
del = MethodStream;
del = MethodString;
// becouse both streamWriter and stringwriter
// inherit from Textwriter
// you can use the covarianceDel with both methods
```

Contravariance with delegates

```
void DoSomething(TextWriter tw) { }
public delegate void ContravarianceDel(StreamWriter tw);

ContravarianceDel del = DoSomething;

// Because the method DoSomething can work with a
   TextWriter, it surely can also work with
// a StreamWriter.Because of contravariance, you can call
   the delegate and pass an instance of
// StreamWriter to the DoSomething method.
```

Лямбда выражения

```
Calculate calc = (x, y) => x + y;
Console.WriteLine(calc(3, 4)); //
  Displays 7

calc = (x, y) => x * y;
Console.WriteLine(calc(3, 4)); //
  Displays 12
```

Многострочные лямбда-выражения

```
Calculate calc =
(x, y) =>
{
  Console.WriteLine("Adding numbers");
  return x + y;
};
int result = calc(3, 4);
// Displays
// Adding numbers
```

Встроенные делегаты

```
public delegate int Calculate(int x, int y);
Func<int, int, int>

Action<int, int> calc = (x, y) =>
{
  Console.WriteLine(x + y);
};
calc(3, 4); // Displays 7
```

Publish-Subscribe (pub-sub)

```
public class Pub
{
  public Action OnChange { get; set; }
  public void Raise()
  {
    if (OnChange != null)
    {
      OnChange();
    }
  }
}
```

Publish-Subscribe (pub-sub)

```
public void CreateAndRaise()
{
  Pub p = new Pub();

  p.OnChange += () => Console.WriteLine("Event raised to
   method 1");

  p.OnChange += () => Console.WriteLine("Event raised to
   method 2");

  p.Raise();
}
```

Недостатки предыдущей схемы

1) Если пользователь класса ошибется и введет = вместо +=, то он затрет все предыдущие значения в OnChange():

```
p.OnChange = () => Console.WriteLine("Event raised to
   method 2");
```

2) любой может вызвать OnChange() напрямую (`p.OnChange()` – не через `p.Raise()`)

Поэтому нужно использовать event-ы

События(events)

```
public class Pub
{
  public event Action OnChange = delegate { };
  public void Raise()
  {
    OnChange();
  }
}
```

Слово events: преимущества

- An *event cannot be directly assigned to (with the = instead of +=) operator.*
- Another change is that no outside users can raise your *event. It can be raised only by code that's part of the class that defined the event.*
- Listing also uses some special syntax to initialize the *event to an empty delegate. This way, you can remove the null check around raising the event because you can be certain that the event is never null.*

EventHandler u EventHandler<T>

- There is, however, one change you still have to make to follow the coding conventions in the .NET Framework. Instead of using the *Action type for your event, you should use the EventHandler or EventHandler<T>. EventHandler is declared as the following delegate:*
- public delegate void EventHandler(object sender, EventArgs e);

Пример

```
public class MyArgs : EventArgs
{
  public MyArgs(int value)
  {
    Value = value;
  }
  public int Value { get; set; }
}
```

Пример

```
public class Pub
{
  public event EventHandler<MyArgs> OnChange =
                                  delegate { };
  public void Raise()
  {
    OnChange(this, new MyArgs(42));
  }
}
```

Пример(используем event-ы)

```
public void CreateAndRaise()
{
  Pub p = new Pub();
  p.OnChange += (sender, e) =>
        Console.WriteLine("Event raised: {0}",
                                    e.Value);
  p.Raise();
}
```

custom event accessor

```
public class Pub
{
  private event EventHandler<MyArgs> onChange =
                                              delegate { };
  public event EventHandler<MyArgs> OnChange
  {
    add
    {
      lock (onChange)
      {
        onChange += value;
      }
```

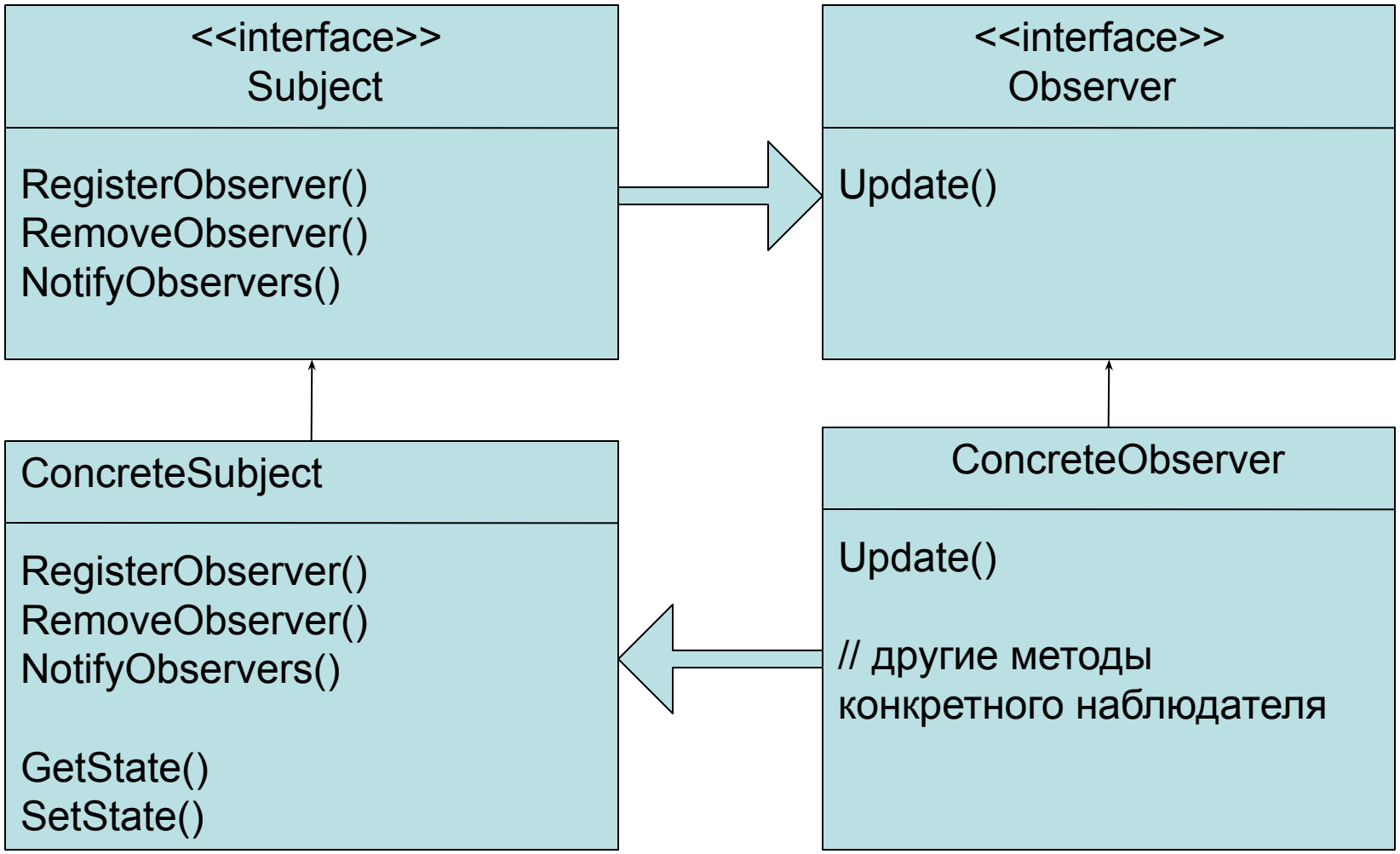
custom event accessor

```
    }
    remove
    {
      lock (onChange)
      {
        onChange -= value;
      }
    }
  }
  public void Raise()
  {
    onChange(this, new MyArgs(42));
  }
}
```

custom event accessor

```
public void CreateAndRaise()
{
  Pub p = new Pub();
  p.OnChange += (sender, e) =>
        Console.WriteLine("Event raised: {0}",
                                    e.Value);
  p.Raise();
}
// код пользователя не изменился
```

Наблюдатель (Observer)



Интерфейс Publisher

```
interface Publisher
{
  void RegisterObserver(Observer o);
  void RemoveObserver(Observer o);
  void NotifyObservers();
}
```

Класс Observer

```
interface Observer
{
  void Update(double dollar, double euro,
  double poundSterling);
}
```

Класс ConcretePublisher

```
class ConcretePublisher : Publisher
    {
        private List<Observer> observers;

        public double Dollar { private get; set; }
        public double Euro { private get; set; }
        public double PoundSterling { private get; set; }

        public void Set(double d, double e, double ps)
        {
            Dollar = d;
            Euro = e;
            PoundSterling = ps;
        }
```

Добавление/удаление наблюдателей

```
public void RegisterObserver(Observer o)
{
    observers.Add(o);
}

public void RemoveObserver(Observer o)
{
    int i = observers.IndexOf(o);
    if (i >= 0)
    {
        observers.RemoveAt(i);
    }
}
```

Оповещение наблюдателей

```
public void NotifyObservers()
{
  for (int i = 0; i < observers.Count; i++)
  {
    observers[i].Update(Dollar, Euro,
                        PoundSterling);
  }
}
```

Конструктор ConcretePublisher

```
public ConcretePublisher(double d, double e,
  double ps)
        {
            Dollar = d;
            Euro = e;
            PoundSterling = ps;

            observers = new List<Observer>();
        }
    }
```

Класс ConcreteObserver

```
class ConcreteObserver : Observer
{
  private string name;
  public ConcreteObserver(string n) {
    name = n;
  }
  public void Update(double dollar, double euro, double
   poundSterling) {
    Console.WriteLine("{0} dollar = {1}", name, dollar);
    Console.WriteLine("{0} euro = {1}", name, euro);
    Console.WriteLine("{0} pound sterling = {1}", name,
                                          poundSterling);
  }
}
```

Метод Main

```
ConcretePublisher bloomberg = new ConcretePublisher(60,
   70, 100);
ConcreteObserver Jhon = new ConcreteObserver("Jhon");
ConcreteObserver Dafna = new ConcreteObserver("Dafna");
ConcreteObserver Dave = new ConcreteObserver("Dave");

bloomberg.RegisterObserver(Jhon);
bloomberg.RegisterObserver(Dafna);
bloomberg.RegisterObserver(Dave);

bloomberg.NotifyObservers();
```

Вывод

Jhon dollar = 60

Jhon euro = 70

Jhon pound sterling = 100

Dafna dollar = 60

Dafna euro = 70

Dafna pound sterling = 100

Dave dollar = 60

Dave euro = 70

Dave pound sterling = 100

Метод Main

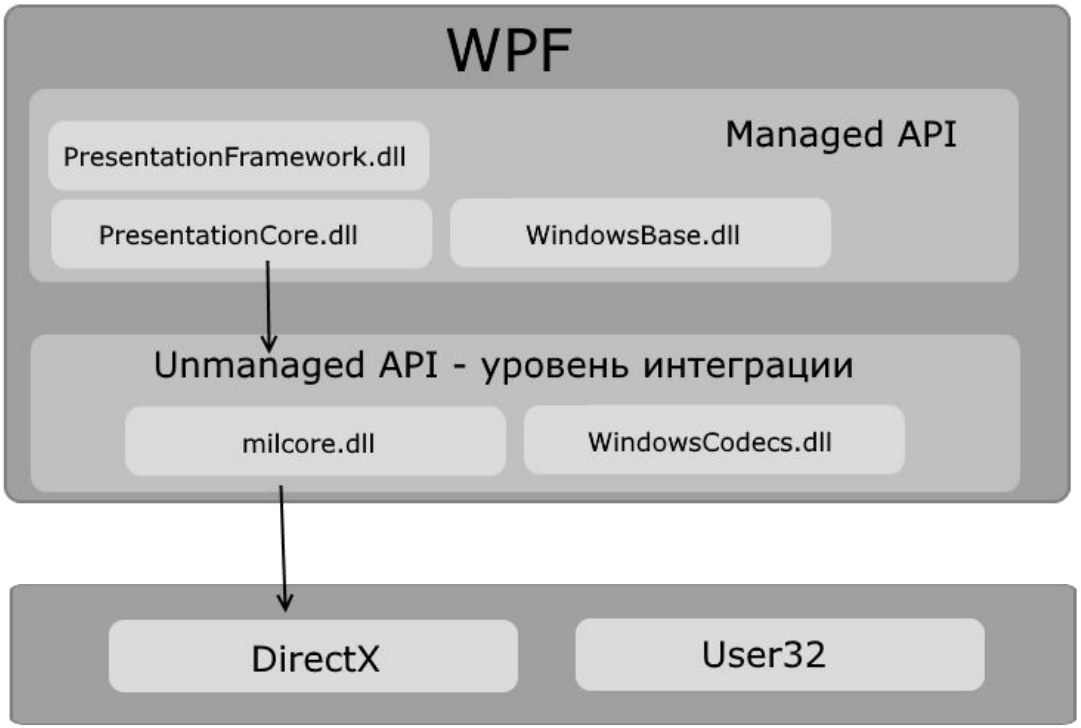
```
Console.WriteLine();
Console.WriteLine("After Brexit:");
bloomberg.PoundSterling = 86;
bloomberg.RemoveObserver(Dave);
Console.WriteLine();

bloomberg.NotifyObservers();
```

Вывод

After Brexit:

Jhon dollar = 60
Jhon euro = 70
Jhon pound sterling = 86
Dafna dollar = 60
Dafna euro = 70
Dafna pound sterling = 86

Windows Presentation Foundation (WPF). Общие сведения



Введение в язык XAML

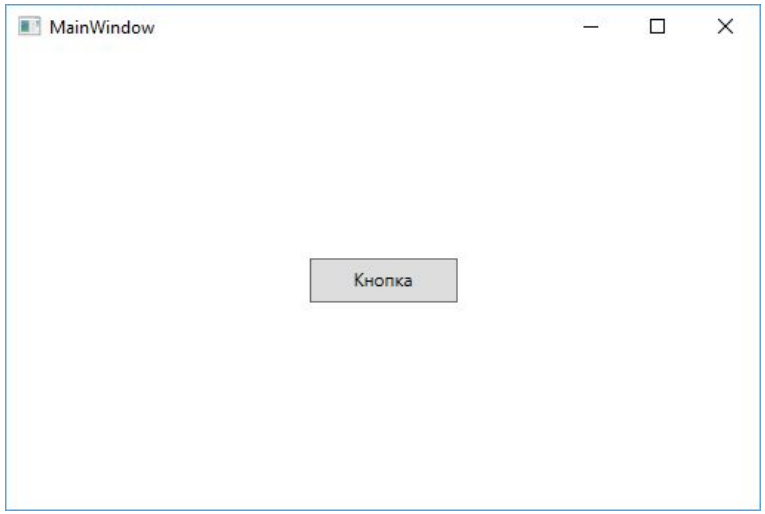
```
<Window x:Class="WpfApp2.MainWindow"
xmlns="http://schemas.microsoft.com/winfx/2006/xaml/
presentation"
xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility
   /2006"
xmlns:local="clr-namespace:WpfApp2"
mc:Ignorable="d"
Title="MainWindow" Height="350" Width="525">
    <Grid>

    </Grid>
</Window>
```

- Можно так: <Window></Window>
- А можно так: <Window />
- Т. е. очень похоже на HTML(или XML)
- Но в отличие от XML элементу может соответствовать некоторый класс (например Button)

Окно с кнопкой

```
<Window x:Class="WpfApp2.MainWindow"
   xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentat
   ion"
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:WpfApp2"
mc:Ignorable="d"
Title="MainWindow" Height="350" Width="525">
 <Grid x:Name="grid1">
    <Button x:Name="button1"  Width="100" Height="30" Content="
   Кнопка" />
 </Grid>
</Window>
```

Окно с кнопкой



Специальные символы

<Button Content="<"Hello">" />

Символ	Код
<	<
>	>
&	&
"	"

<Button Content="<"Hello">" />

xml:space="preserve"

```
<Button>
    Hello          World
</Button>
```

```
<Button xml:space="preserve">
   Hello       World
</Button>
```

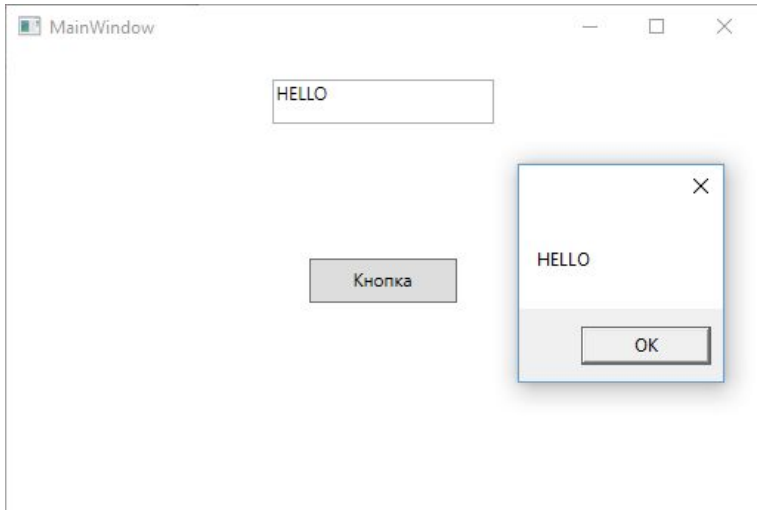
Обработка событий по кнопке

```
<Window x:Class="WpfApp2.MainWindow"
        .........
        Title="MainWindow" Height="350" Width="525">
  <Grid x:Name="grid1">
    <TextBox x:Name="textBox1" Width="150" Height="30"
   VerticalAlignment="Top" Margin="20" />
    <Button x:Name="button1"  Width="100" Height="30"
   Content="Кнопка" Click="Button_Click" />
  </Grid>
</Window>
```

Обработка событий по кнопке

```
public partial class MainWindow : Window
{
  public MainWindow()
  {
    InitializeComponent();
  }
  private void Button_Click(object sender, RoutedEventArgs e)
  {
    string text = textBox1.Text;
    if (!text.Equals(""))
    {
      MessageBox.Show(text);
    }
  }
}
```

Обработка событий по кнопке



Простейшие свойства

- Name
- Width
- Height
- Content
- HorizontalAlignment
- VerticalAlignment
- Background
- Margin
- Panel.Index
- …

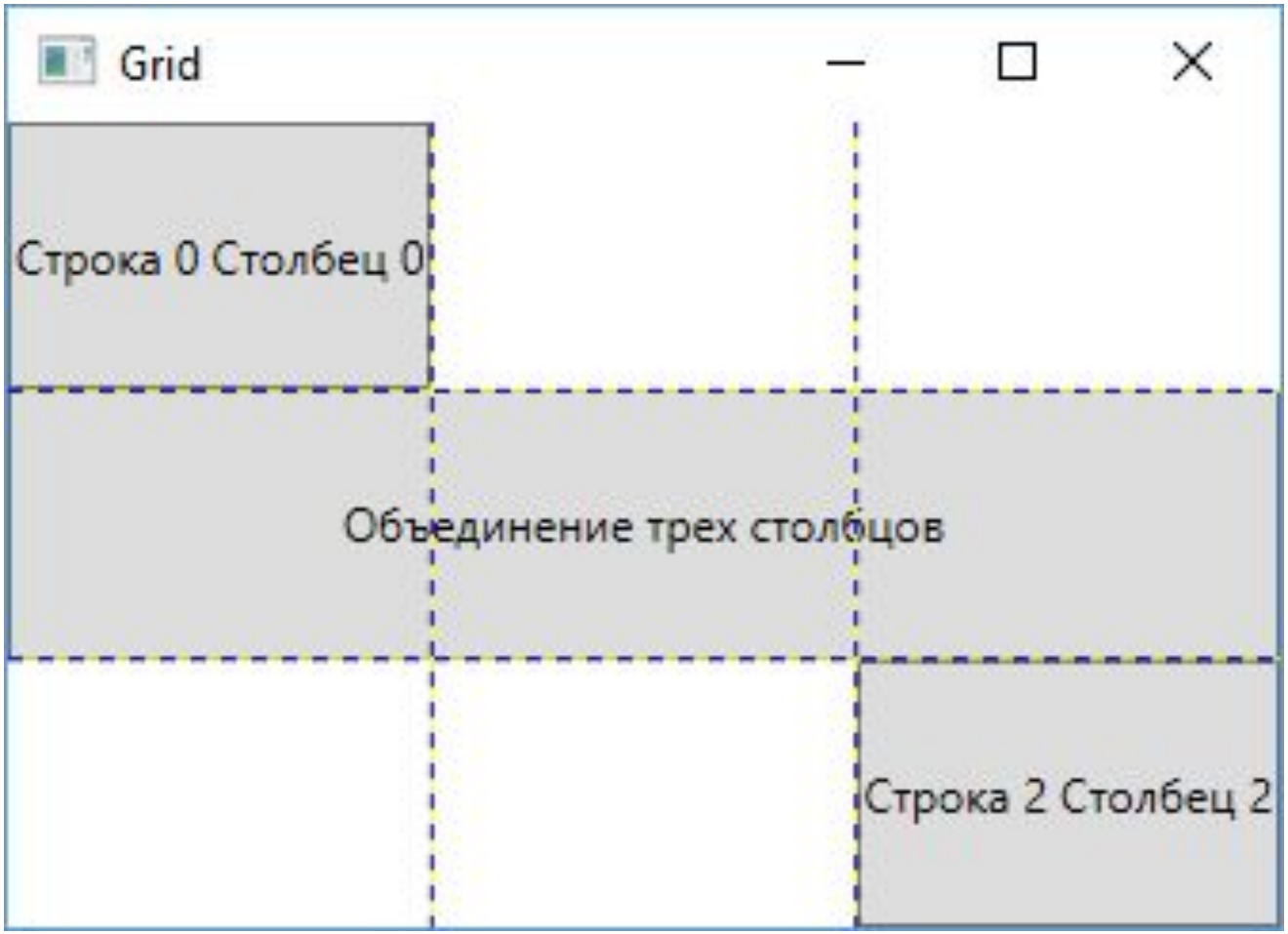
Компоновка

- Grid
- UniformGrid
- GridSplitter
- StackPanel
- DockPanel
- WrapPanel
- Canvas

Grid

```
<Window x:Class="WpfApp2.MainWindow"
  Title="MainWindow" Height="350" Width="525">
    <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>
```

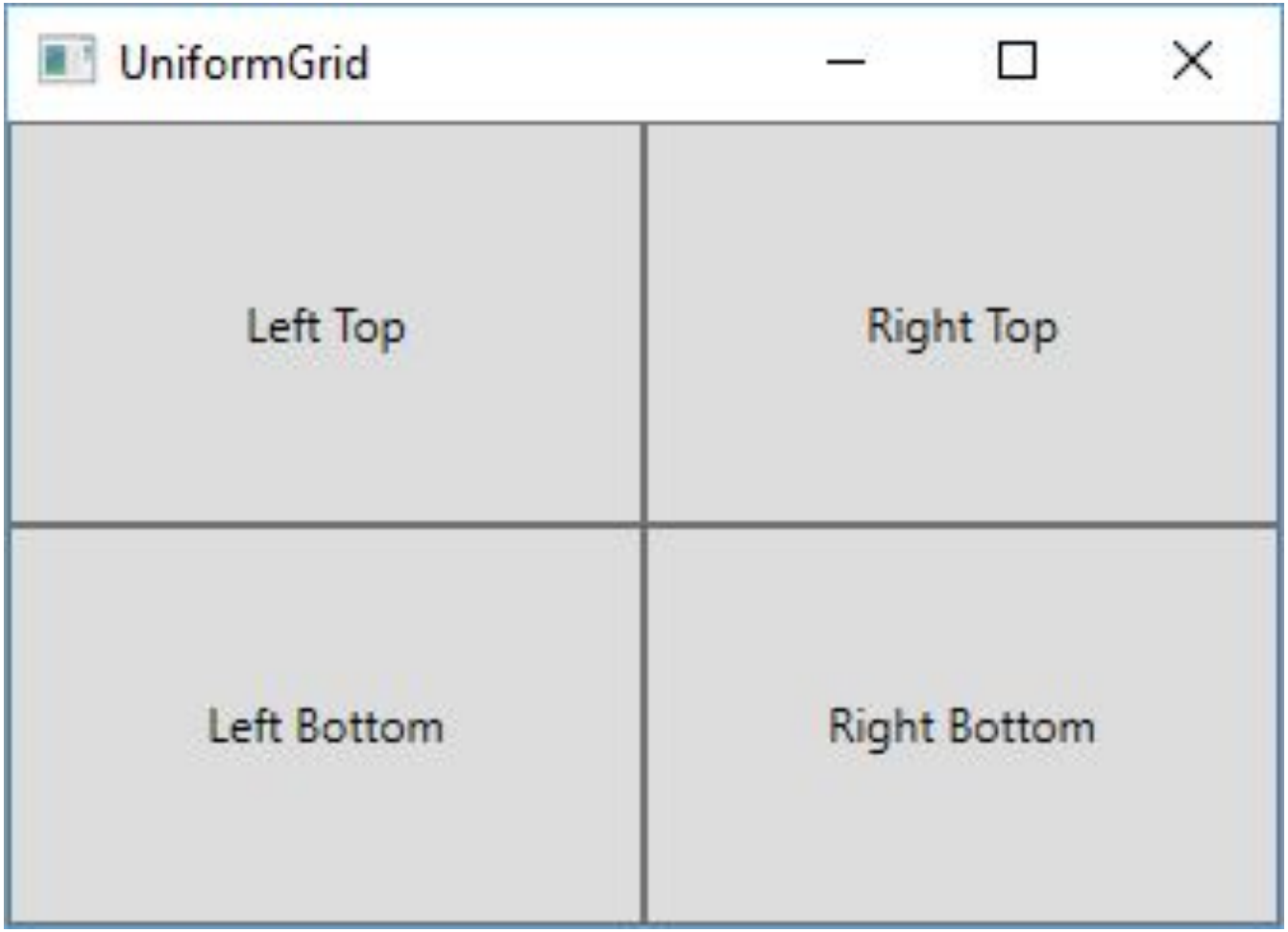
Grid


Grid
Строка 0 Столбец 0
Объединение трех столбцов
Строка 2 Столбец 2

UniformGrid

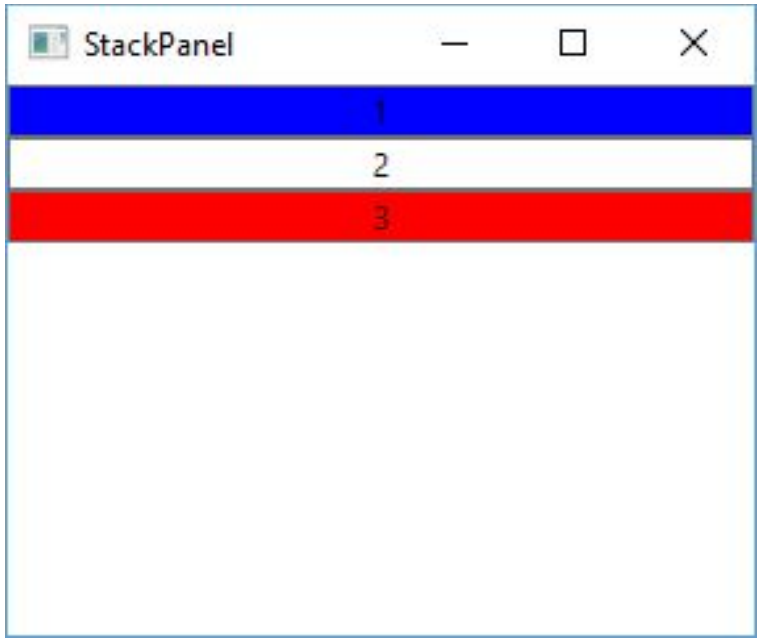
```
<UniformGrid Rows="2" Columns="2">
        <Button Content="Left Top" />
        <Button Content="Right Top" />
        <Button Content="Left Bottom" />
        <Button Content="Right Bottom" />
    </UniformGrid>
```

UniformGrid



StackPanel

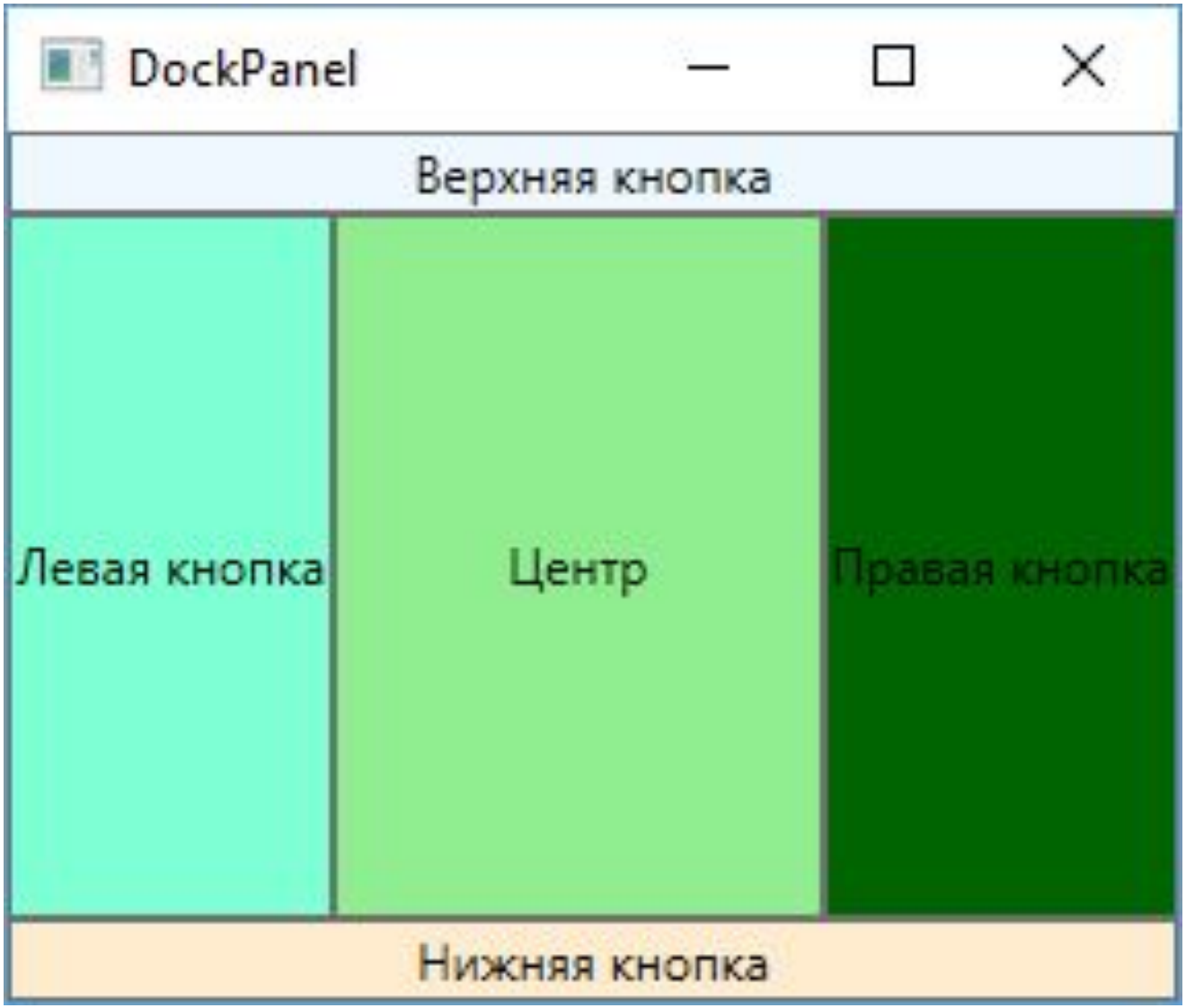
```
<StackPanel>
        <Button Background="Blue" Content="1" />
        <Button Background="White" Content="2" />
        <Button Background="Red" Content="3" />
</StackPanel>
```



DockPanel

```
<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>
```

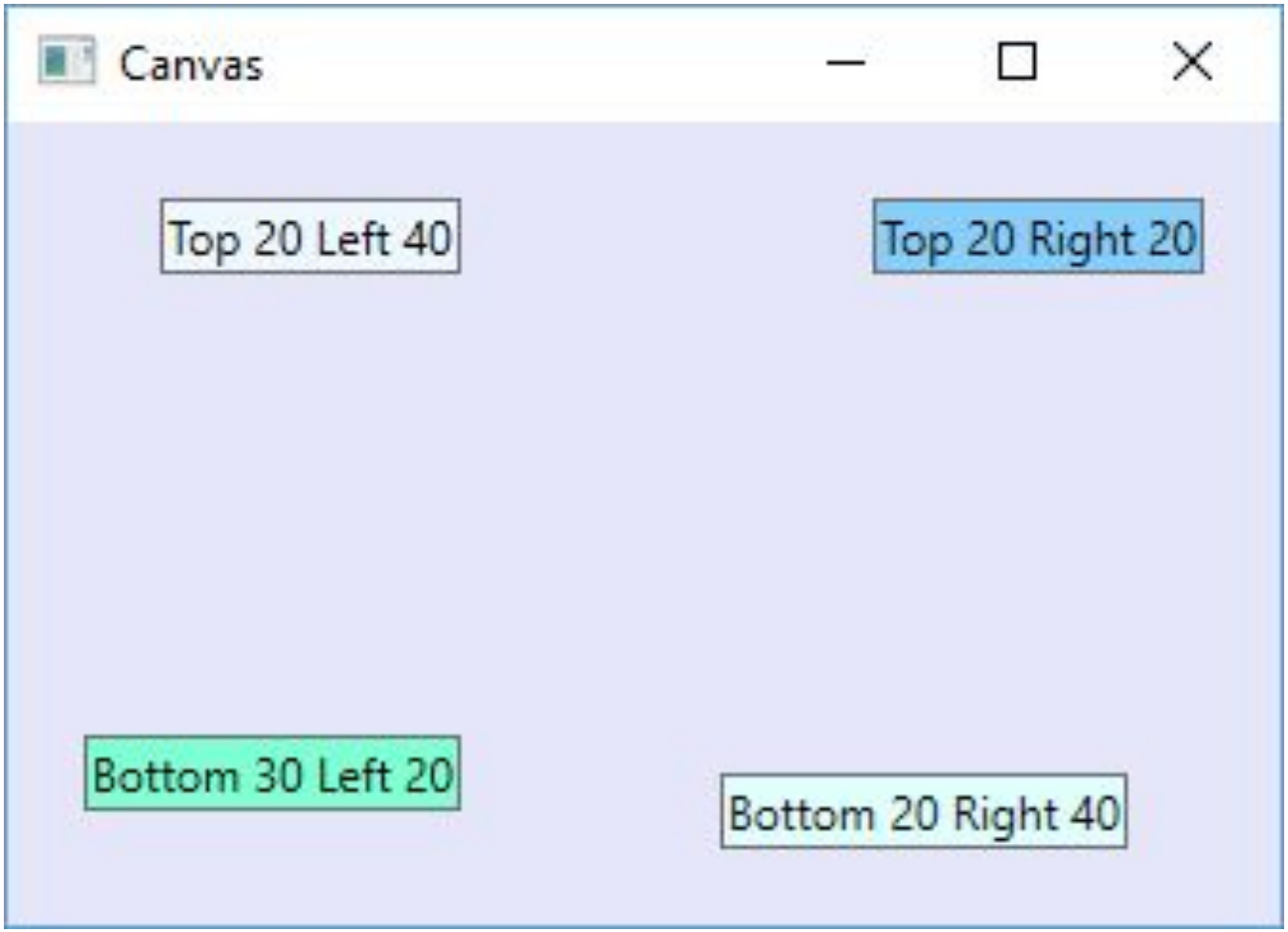
DockPanel


DockPanel
Верхняя кнопка
Левая кнопка
Центр
Правая кнопка
Нижняя кнопка

Canvas

```
<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>
```

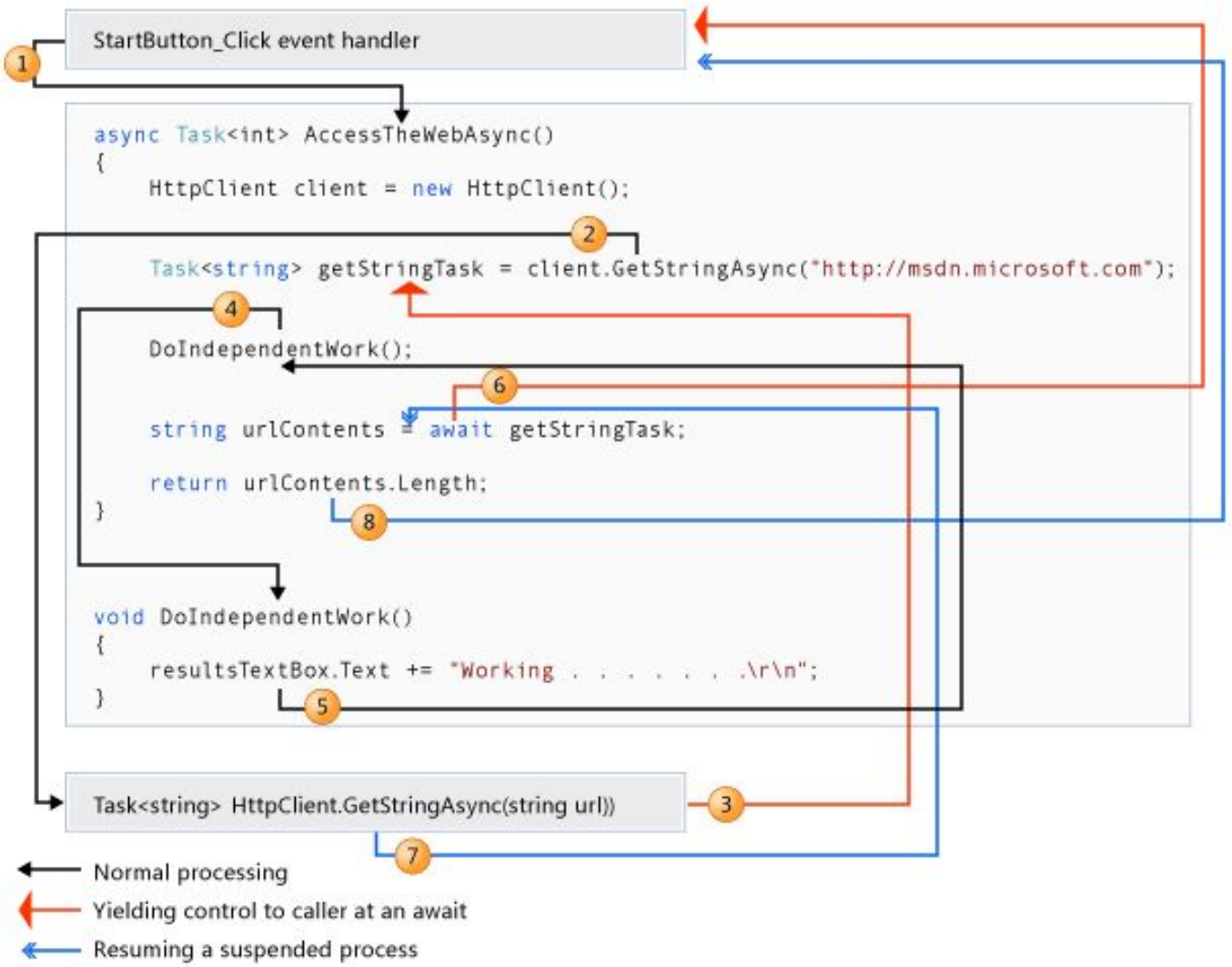
Canvas


Canvas
Top 20 Left 40
Top 20 Right 20
Bottom 30 Left 20
Bottom 20 Right 40

Async и await

```
async Task<int> AccessTheWebAsync() {

  HttpClient client = new HttpClient();

  Task<string> getStringTask =
      client.GetStringAsync("http://msdn.microsoft.com");

  DoIndependentWork();

  string urlContents = await getStringTask;

  return urlContents.Length;
}
```

Async и await



Пример

```
<Window x:Class="WpfApp34.MainWindow"
…
        Title="MainWindow" Height="207.014" Width="301.471">
    <Grid>
        <Button Content="Button" HorizontalAlignment="Left"
           Margin="34,24,0,0" VerticalAlignment="Top" Width="75"
           Click="Button_Click"/>
    </Grid>
</Window>
```

Пример

```
public partial class MainWindow : Window
    {
        public MainWindow() {
            InitializeComponent();
        }

        private void Button_Click(object sender,
   RoutedEventArgs e) {
            var b = sender as Button;
            b.IsEnabled = false;
            Run();
            b.IsEnabled = true;
        }
```

Пример

```
    void Run() {
        Task.Delay(2000).Wait();
    }
}
```

Решение 1

```
public partial class MainWindow : Window
    {
        public MainWindow()
        {
            InitializeComponent();
        }
        private void Button_Click(object sender, RoutedEventArgs e)
        {
            var b = sender as Button;

            b.IsEnabled = false;

            var t = new Thread(Run);
            t.IsBackground = true;
            t.Start();

            b.IsEnabled = true;
        }
```

Решение 1

```
    void Run()
    {
        Task.Delay(2000).Wait();
    }
}
```

Ошибочный пример

```
public partial class MainWindow : Window
{
  private void Button_Click(object sender,
   RoutedEventArgs e)
  {
    var b = sender as Button;
    b.IsEnabled = false;
    var t = new Thread(() => {
        Run();
        b.IsEnabled = true;
     });
     t.IsBackground = true;
     t.Start();
   }
```

Ошибочный пример

```
  void Run()
  {
    Task.Delay(2000).Wait();
  }
}
```

Асинхронное решение

```
public partial class MainWindow : Window
{
    private async void Button_Click(object sender,
              RoutedEventArgs e)
    {
      var b = sender as Button;
      b.IsEnabled = false;
      await Run();
      b.IsEnabled = true;
    }

    async Task Run()
    {
      await Task.Delay(2000);
    }
}
```

Для библиотечных функций

```
public partial class MainWindow : Window
{
  private async void Button_Click(object
                    sender, RoutedEventArgs e)
  {
    var b = sender as Button;
    b.IsEnabled = false;
    await Task.Factory.StartNew(Run);
    b.IsEnabled = true;
  }
```

Для библиотечных функций

```
  void Run()
  {
    Task.Delay(2000).Wait();
  }
}
```

Возвращение значения

```
public partial class MainWindow : Window
{
  private async void Button_Click(object sender,
                        RoutedEventArgs e)
  {
    int res = 0;
    output.Text = res.ToString();
    var b = sender as Button;
    b.IsEnabled = false;
    res = await Run();
    output.Text = res.ToString();
    b.IsEnabled = true;
  }
```

Возвращение значения

```
  Task<int> Run()
  {
    return Task.Run(() =>
    {
      Task.Delay(2000).Wait();
      return 6;
    });
  }
}
```