.NET & WPF

윈도우즈 프로그래밍

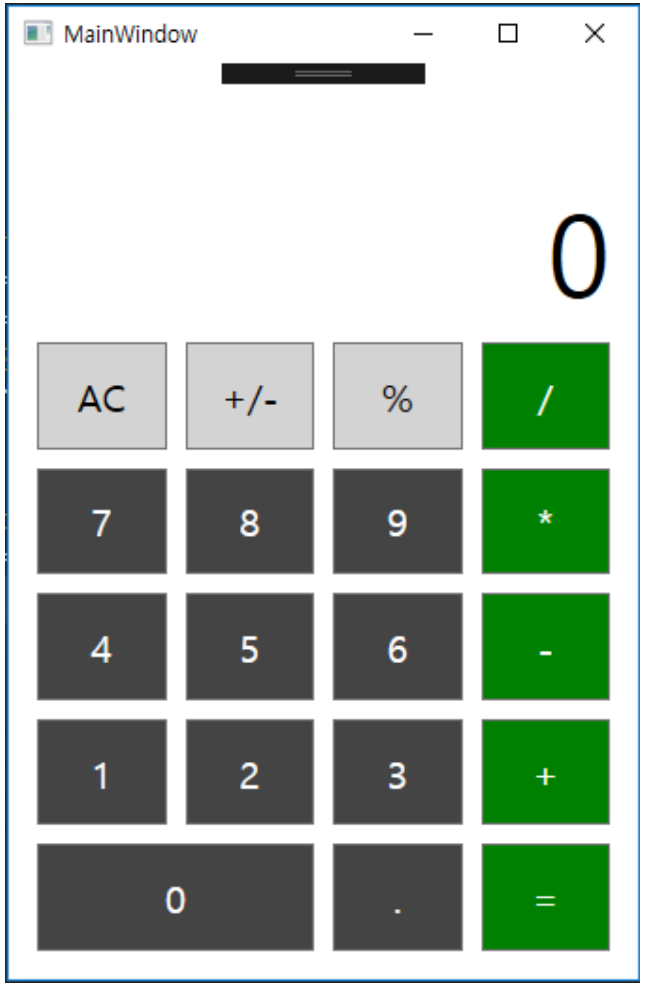
김 형 기

hk.kim@jbnu.ac.kr

계산기

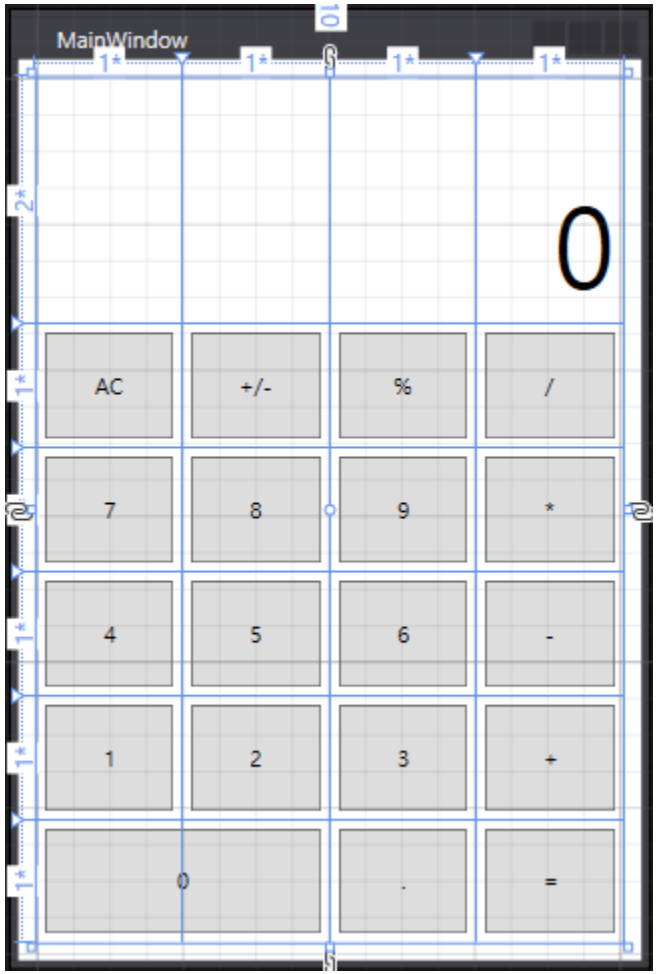
계산기 프로그램 개요

- 기본적인 컨트롤과 Code-behind 작성으로 계산기를 구현
- 주요 내용
  - Grid의 사용 방법
  - 컨트롤 객체의 코드 접근 방법
  - 이벤트 핸들러의 생성과 사용
  - Enum, Class 정의 방법
  - 태그 확장 문법의 이해
  - 스타일링



계산기 프로그램

- 1. 계산기 레이아웃 생성
  - Grid를 활용한 격자 생성
    - Row & Column definition
    - Star sizing
  - Label, Button
    - Margin
    - FontSize
    - Alignment



계산기 프로그램

- 2. Code-behind
  - 특정 element에 x:Name으로 객체 이름을 부여
  - Code-behind에서는 객체 이름을 통해 Property에 접근
  - 나머지 Element들에도 적절한 이름을 부여

```
<Label Content="0"
               Grid.ColumnSpan="4"
               HorizontalAlignment="Right"
               VerticalAlignment="Bottom"
               FontSize="60"
               x:Name="resultLabel"/>
```

```
public MainWindow()
        {
            InitializeComponent();

            resultLabel.Content = "12345";
        }
```

계산기 프로그램

- 3. 이벤트 핸들러 생성
  - Button의 Click 이벤트 핸들러 생성
  - 적절한 새 이벤트 이름을 IDE에서 자동 추천해 주며, Method 블록이 Code-behind에 자동 생성

```
<Button Content="7"
                x:Name="sevenButton"
                Grid.Row="2"
                Grid.Column="0"
                Margin="5"
                Click="sevenButton_Click"/>
```

```
private void sevenButton_Click(object sender, RoutedEventArgs e)
        {
            if (resultLabel.Content.ToString() == "0")
            {
                resultLabel.Content = "7";
            }
            else
            {
                resultLabel.Content = $"{resultLabel.Content}7";

                //or
                //resultLabel.Content += "7";
            }
        }
```

계산기 프로그램

- 4. 코드를 통한 이벤트 핸들러 생성
  - AC, +/-, % 버튼에 대한 이벤트 핸들러를 생성자에서 생성
  - +/- 버튼의 동작에 사용되는 method들
    - Type.TryParse(), Object.ToString()
  - Out 키워드의 의미

```
acButton.Click += AcButton_Click;
negativeButton.Click += NegativeButton_Click;
percentButton.Click += PercentButton_Click;
equalButton.Click += EqualButton_Click;
```

```
private void NegativeButton_Click(object sender, RoutedEventArgs e)
        {
            //Tryparse 메소드, ToString() 메소드, out 키워드의 사용법 숙지
            if (double.TryParse(resultLabel.Content.ToString(), out lastNumber))
            {
                lastNumber = lastNumber * -1;
                resultLabel.Content = lastNumber.ToString();
            }
        }
```

계산기 프로그램

- 5. NumberButton_Click 이벤트 핸들러
  - 버튼에 대해 각각의 이벤트 핸들러를 XAML에서 개별적으로 만들어 주는 것은 비효율적
    - ➢ 0~9까지 10번의 중복적인 작업 필요
  - 버튼 이벤트 핸들러의 sender를 활용
  - OperationButton_Click도 유사하게 구현 가능

이름	값
e	{System.Windows.RoutedEventArgs}
selectedValue	0
sender	{System.Windows.Controls.Button: 7}
this	{Calculator.MainWindow}

계산기 프로그램

- 6. OperationButton_Click 이벤트 핸들러
  - Operation을 저장할 enum 타입 생성
  - 계산을 수행할 SimpleMath 클래스 생성
    - 계산인 equal button을 눌렀을 때 수행
    - 객체를 생성하지 않고 메소드를 호출하기 위해 static 선언
  - Point button click 핸들러 추가

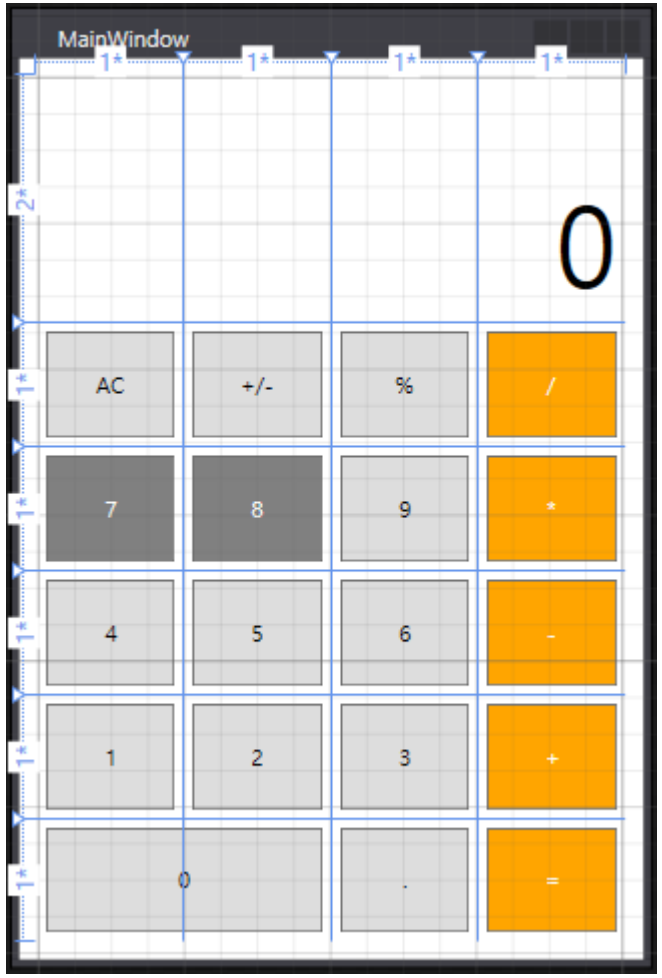
계산기 프로그램

- 7. 테스트 및 수정
  - 계산 기능 테스트
  - 예외 Case? → Division by 0
    - MessageBox를 통한 처리 예시

```
public static double Div(double n1, double n2)
        {
            if (n2 == 0)
            {
                MessageBox.Show("0으로 나눌 수 없습니다.", "Wrong Operation",
                    MessageBoxButton.OK, MessageBoxImage.Error);
                return 0;
            }
            return n1 / n2;
        }
```

계산기 프로그램

- 8. Basic Styling
  - Background, Foreground Attribute
    - ➢ Hexadecimal RGB
  - 여러 Element의 색상을 바꾸고 싶다면?
    - ➢ 효율적인 스타일 관리 방법이 필요



계산기 프로그램

- 9. Static Resources
  - 리소스 : 프로그램 내에서 재 사용할 수 있는 객체
  - <Window.Resources> : Window 내에서 사용할 리소스를 정의하는 태그
  - <SolidColorBrush> : 실제로 재사용하는 객체, 접근을 위한 x:Key가 필요
  - **<u>태그 확장 문법 : { }, 상수가 아닌 참조 값을 사용하기 위한 문법</u>**

```
<Window.Resources>
        <SolidColorBrush x:Key="numbersColor"
                         Color="#444444"/>
        <SolidColorBrush x:Key="operatorsColor"
                         Color="Orange"/>
    </Window.Resources>
```

```
<Button Content="/" x:Name="divButton" Grid.Row="1" Grid.Column="3" Margin="5"
            Click="OperationButton_Click"
            Background="{StaticResource operatorsColor}"
            Foreground="White"/>
<Button Content="*" x:Name="mulButton" Grid.Row="2" Grid.Column="3" Margin="5"
            Click="OperationButton_Click"
            Background="{StaticResource operatorsColor}"
            Foreground="White"/>
```

계산기 프로그램

- 10. Application-wide Resources
  - App.xaml : 전역 scope와 같은 공간. Application 전체에서 접근할 수 있는 객체를 정의

```
<Application x:Class="Calculator.App"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:local="clr-namespace:Calculator"
             StartupUri="MainWindow.xaml">
    <Application.Resources>
        <SolidColorBrush x:Key="numbersColor"
                         Color="#444444"/>
        <SolidColorBrush x:Key="operatorsColor"
                         Color="Orange"/>
        <SolidColorBrush x:Key="foregroundColor"
                         Color="White"/>
    </Application.Resources>
</Application>
```

계산기 프로그램

- 11. Implicit Style
  - <Style> 과 <Setter> 를 사용해 컨트롤에 적용되는 기본 값을 설정 가능
    - 특정 컨트롤에서 값을 override하여 예외 적용 가능

- 12. Explicit Style
  - <Style>에 key를 부여
  - 이를 적용할 Button의 Style에서 key를 통해 스타일을 명시