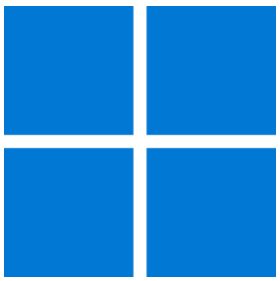
tutorial

Windows App SDK



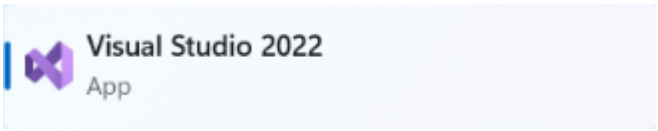
Deal or Not

Deal or Not shows how you can create simple box-opening game where every so often you can choose to accept a deal, or not and continue but risk winning a smaller amount, the last box will be the amount won, unless you took the deal, using a toolkit from **NuGet** using the **Windows App SDK**.

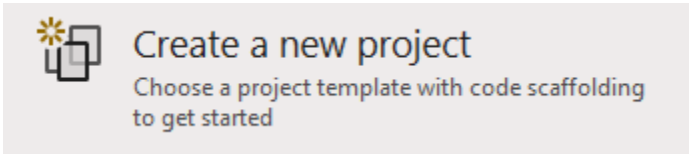
Step 1

Follow **Setup and Start** on how to get **Setup** and **Install** what you need for **Visual Studio 2022** and **Windows App SDK**.

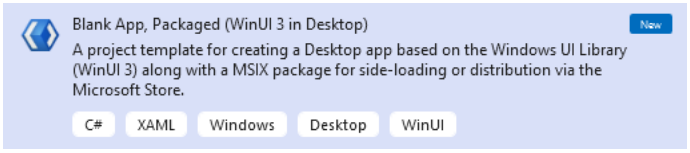
In **Windows 11** choose **Start** and then find or search for **Visual Studio 2022** and then select it.



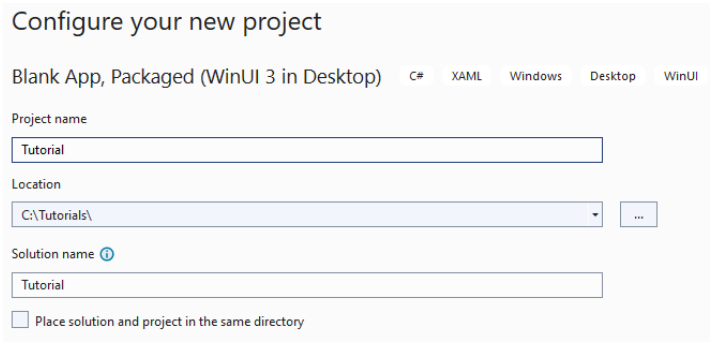
Once **Visual Studio 2022** has started select **Create a new project**.



Then choose the **Blank App, Packages (WinUI in Desktop)** and then select **Next**.

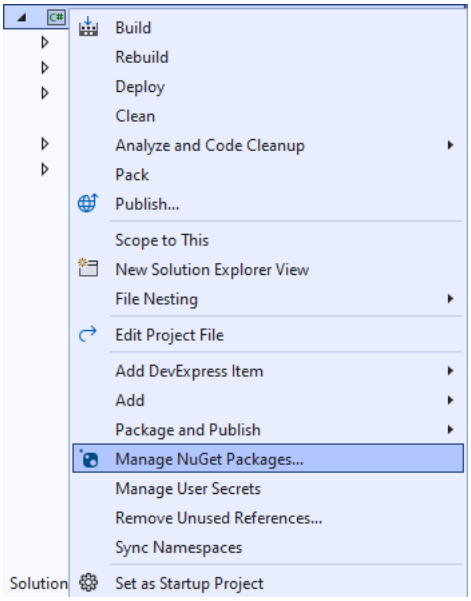


After that in **Configure your new project** type in the **Project name** as *DealOrNot*, then select a Location and then select **Create** to start a new **Solution**.



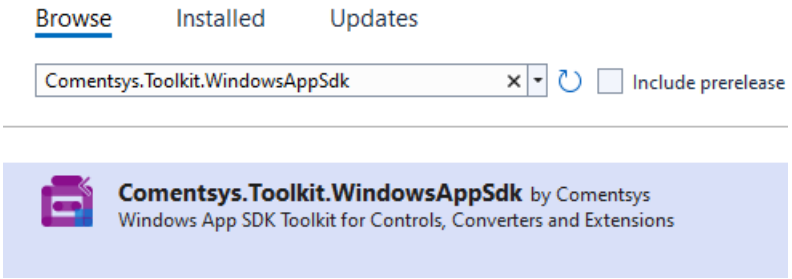
Step 2

Then in **Visual Studio** within **Solution Explorer** for the **Solution**, right click on the **Project** shown below the **Solution** and then select **Manage NuGet Packages...**



Step 3

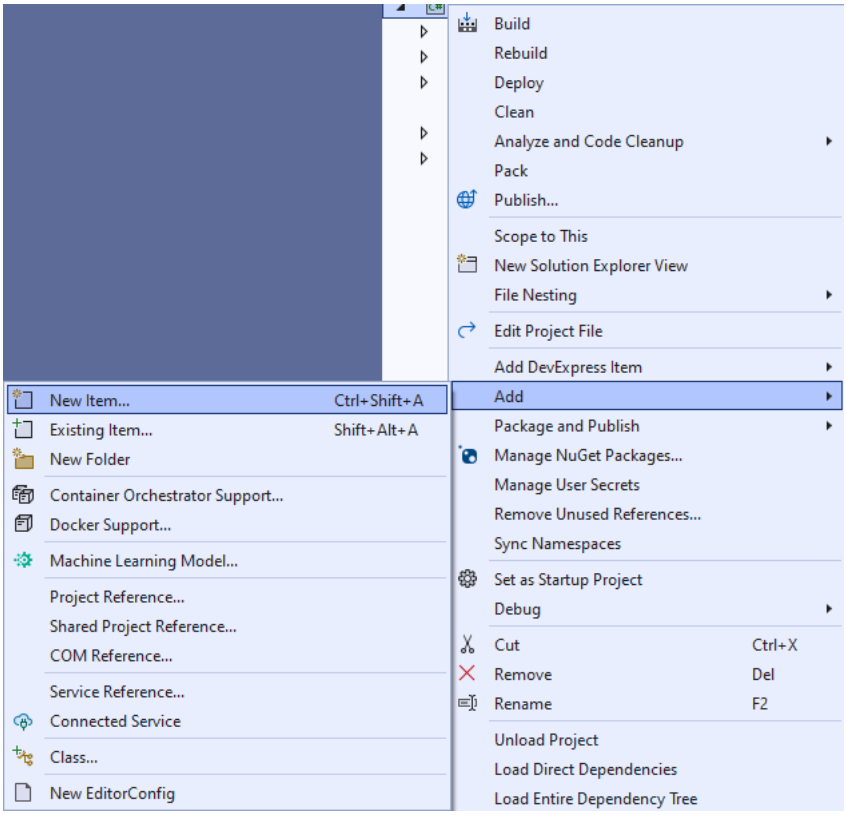
Then in the **NuGet Package Manager** from the **Browse** tab search for **Comentsys.Toolkit.WindowsAppSdk** and then select **Comentsys.Toolkit.WindowsAppSdk by Comentsys** as indicated and select **Install**



This will add the package for **Comentsys.Toolkit.WindowsAppSdk** to your **Project**. If you get the **Preview Changes** screen saying **Visual Studio is about to make changes to this solution. Click OK to proceed with the changes listed below.** You can read the message and then select **OK** to **Install** the package,, then you can close the **tab** for **Nuget: DealOrNot** by selecting the **x** next to it.

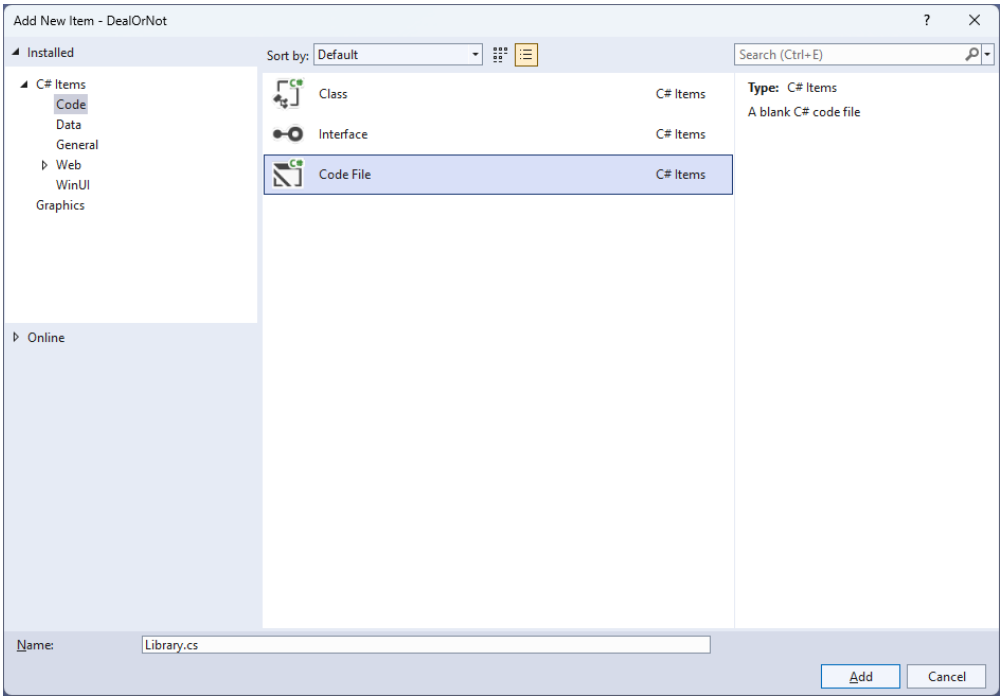
Step 4

Then in **Visual Studio** within **Solution Explorer** for the **Solution**, right click on the **Project** shown below the **Solution** and then select **Add** then **New Item...**



Step 5

Then in **Add New Item** from the **C# Items** list, select **Code** and then select **Code File** from the list next to this, then type in the name of *Library.cs* and then **Click** on **Add**.



Step 6

You will now be in the **View** for the **Code** of *Library.cs*, within this first type the following **Code**:

```
using Comentsys.Toolkit.WindowsAppSdk;
using Microsoft.UI;
using Microsoft.UI.Text;
using Microsoft.UI.Xaml;
using Microsoft.UI.Xaml.Controls;
using Microsoft.UI.Xaml.Media;
using Microsoft.UI.Xaml.Shapes;
using System;
using System.Collections.Generic;
using System.Globalization;
using System.Linq;
using Windows.UI;

public class Library
{
    private const string title = "Deal or Not";
    private const int rate = 5;
    private static readonly double[] amounts =
    {
        0.01, 0.10, 0.50, 1, 5, 10, 50, 100, 250, 500, 750,
        1000, 3000, 5000, 10000, 15000, 20000, 35000, 50000, 75000, 100000, 250000
    };
    private static readonly string[] colors =
    {
        "0026ff", "0039ff", "004dff", "0060ff", "0073ff", "0086ff",
        "0099ff", "0099ff", "0099ff", "00acff", "00bfff",
        "ff5900", "ff4d00", "ff4000", "ff3300", "ff2600", "ff2600",
        "ff2600", "ff2600", "ff1a00", "ff1c00", "ff0d00",
    };
    private static readonly string[] names = {
        "a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k",
        "l", "m", "n", "o", "p", "q", "r", "s", "t", "u", "v"
    };
    private readonly Random _random = new((int)DateTime.UtcNow.Ticks);
    private readonly List<double> _values = new();

    private int _turn;
    private bool _over;
    private bool _dealt;
    private double _amount;
    private Dialog _dialog;

    // Choose, Get Color, Get Background & Get Amount

    // Get Offer & Select Box

    // Add Box

    // Add Row, Layout & New
}
```

Class defined so far *Library.cs* has **using** for package of **Comentsys.Toolkit.WindowsAppSdk** and others.

Step 7

Still in the **Class** for *Library.cs* after the **Comment** of **// Choose, Get Color, Get Background & Get Amount** type the following **Methods**:

```
private List<int> Choose(int minimum, int maximum, int total) =>
Enumerable.Range(minimum, maximum)
    .OrderBy(r => _random.Next(minimum, maximum))
        .Take(total).ToList();

private Color GetColor(string hex)
{
    byte r = byte.Parse(hex[0..^4], NumberStyles.HexNumber);
    byte g = byte.Parse(hex[2..^2], NumberStyles.HexNumber);
    byte b = byte.Parse(hex[4..^0], NumberStyles.HexNumber);
    return Color.FromArgb(255, r, g, b);
}

private Color GetBackground(double amount)
{
    var position = Array.FindIndex(amounts, a => a.Equals(amount));
    return GetColor(colors[position]);
}

private Grid GetAmount(double value, Color background)
{
    Grid grid = new()
    {
        Background = new SolidColorBrush(background)
    };
    TextBlock text = new()
    {
        Text = string.Format(new CultureInfo("en-GB"), "{0:c}", value),
        HorizontalAlignment = HorizontalAlignment.Center,
        VerticalAlignment = VerticalAlignment.Center,
        Foreground = new SolidColorBrush(Colors.White),
        Margin = new Thickness(10),
        FontSize = 33
    };
    grid.Children.Add(text);
    return grid;
}
```

Choose is used to select a list of randomised numbers, **GetColor** is used to get a **Color** from the hex representation of the colour and **GetBackground** will use this to get the appropriate background colour based on the amount passed in. **GetAmount** will be used to display an amount with an appropriate colour.

Step 8

While still in the **`Class`** for *Library.cs* after the **Comment** of **`// Get Offer & Select Box`** type in the following **Methods** for **`GetOffer`** to generate an offer and **`SelectBox`** used when picking a box.

```
private double GetOffer()
{
    int count = 0;
    double total = 0.0;
    foreach (double value in _values)
    {
        total += value;
        count++;
    }
    double average = total / count;
    double offer = average * _turn / 10;
    return Math.Round(offer, 0);
}

private async void SelectBox(Button button, string name)
{
    if (!_over)
    {
        if (_turn < names.Length)
        {
            button.Opacity = 0;
            _amount = _values[Array.IndexOf(names, name)];
            bool response = await _dialog.ConfirmAsync(
                GetAmount(_amount, GetBackground(_amount)));
            if (response)
            {
                if (!_dealt && _turn % rate == 0 && _turn > 1)
                {
                    double offer = GetOffer();
                    bool accept = await _dialog.ConfirmAsync(
                        GetAmount(offer, Colors.Black), "Deal", "Not");
                    if (accept)
                    {
                        _amount = offer;
                        _dealt = true;
                    }
                }
                _turn++;
            }
        }
        if (_turn == names.Length || _dealt)
            _over = true;
    }
    if (_over)
    {
        object content = _dealt ?
            GetAmount(_amount, Colors.Black) :
            GetAmount(_amount, GetBackground(_amount));
        await _dialog.ConfirmAsync(content, "Game Over", null);
    }
}
```

Step 9

While still in the **`Class`** for *Library.cs* after the **Comment** of **`// Add Box`** type in the following **Method**:

```
private void AddBox(StackPanel panel, string name, int value)
{
    Button button = new()
    {
        Name = $"box.{name}",
        Margin = new Thickness(5)
    };
    button.Click += (object sender, RoutedEventArgs e) =>
        SelectBox((Button)sender, name);
    StackPanel box = new()
    {
        Width = 100
    };
    Rectangle lid = new()
    {
        Height = 10,
        Fill = new SolidColorBrush(Colors.DarkRed)
    };
    Grid front = new()
    {
        Height = 75,
        Background = new SolidColorBrush(Colors.Red)
    };
    Grid label = new()
    {
        Width = 50,
        Background = new SolidColorBrush(Colors.White),
        HorizontalAlignment = HorizontalAlignment.Center,
        VerticalAlignment = VerticalAlignment.Center
    };
    TextBlock text = new()
    {
        TextAlignment = TextAlignment.Center,
        FontWeight = FontWeights.Bold,
        Foreground = new SolidColorBrush(Colors.Black),
        FontSize = 32,
        Text = value.ToString()
    };
    label.Children.Add(text);
    front.Children.Add(label);
    box.Children.Add(lid);
    box.Children.Add(front);
    button.Content = box;
    panel.Children.Add(button);
}
```

AddBox is used to add a box that can be selected when **Clicked** to the game and give it a game-specific appearance using various elements to create the look-and-feel of the box that can be selected.

Step 10

While still in the **`Class`** for *Library.cs* after the **Comment** of **`// Add Row, Layout & New`** type in the following **Methods** of **`AddRow`** which will add a row of boxes for the game, **`Layout`** which will create the look-and-feel for the game and **`New`** which will start a game.

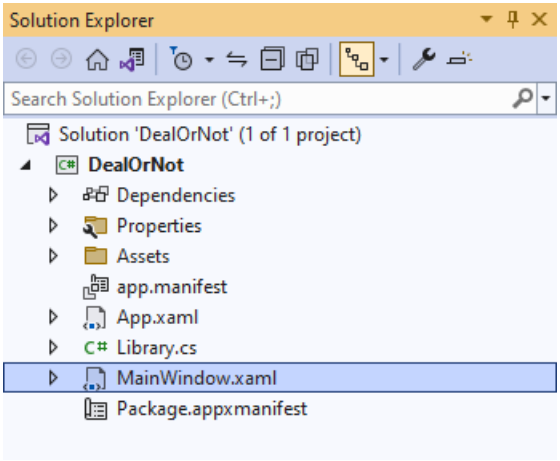
```
private StackPanel AddRow()
{
    int count = 0;
    StackPanel panel = new();
    int[] rows = { 5, 6, 6, 5 };
    for (int r = 0; r < 4; r++)
    {
        StackPanel places = new()
        {
            Orientation = Orientation.Horizontal,
            HorizontalAlignment = HorizontalAlignment.Center
        };
        for (int column = 0; column < rows[r]; column++)
        {
            AddBox(places, names[count], count + 1);
            count++;
        }
        panel.Children.Add(places);
    }
    return panel;
}

private void Layout(Grid grid)
{
    grid.Children.Clear();
    Viewbox view = new()
    {
        Child = AddRow()
    };
    grid.Children.Add(view);
}

public void New(Grid grid)
{
    _turn = 0;
    _amount = 0;
    _over = false;
    _dealt = false;
    _dialog = new Dialog(grid.XamlRoot, title);
    var positions = Choose(0, names.Length, names.Length);
    foreach (var position in positions)
    {
        _values.Add(amounts[position]);
    }
    Layout(grid);
}
```

Step 11

Then from **Solution Explorer** for the **Solution** double-click on **MainWindow.xaml** to see the **XAML** for the **Main Window**.



Step 12

In the **XAML** for **MainWindow.xaml** there be some **XAML** for a **`StackPanel`**, this should be **Removed** by removing the following:

```
<StackPanel Orientation="Horizontal"
HorizontalAlignment="Center" VerticalAlignment="Center">
    <Button x:Name="myButton" Click="myButton_Click">Click Me</Button>
</StackPanel>
```

Step 13

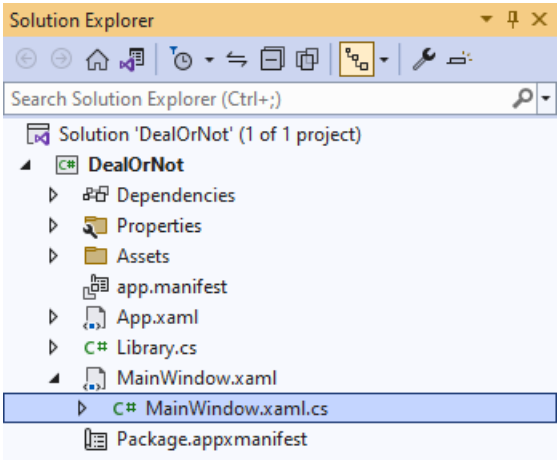
While still in the **XAML** for **MainWindow.xaml** above **`</Window>`**, type in the following **XAML**:

```
<Grid>
    <Viewbox>
        <Grid Margin="50" Name="Display"
        HorizontalAlignment="Center"
        VerticalAlignment="Center" Loaded="New"/>
    </Viewbox>
    <CommandBar VerticalAlignment="Bottom">
        <AppBarButton Icon="Page2" Label="New" Click="New"/>
    </CommandBar>
</Grid>
```

This **XAML** contains a **`Grid`** with a **`Viewbox`** which will scale a **`Grid`**. It has a **`Loaded`** event handler for **`New`** which is also shared by the **`AppBarButton`**.

Step 14

Then, within **Solution Explorer** for the **Solution** select the arrow next to **MainWindow.xaml** then double-click on **MainWindow.xaml.cs** to see the **Code** for the **Main Window**.



Step 15

In the **Code** for **MainWindow.xaml.cs** there be a **Method** of **`myButton_Click(...)`** this should be **Removed** by removing the following:

```
private void myButton_Click(object sender, RoutedEventArgs e)
{
    myButton.Content = "Clicked";
}
```

Step 16

Once **`myButton_Click(...)`** has been removed, type in the following **Code** below the end of the **Constructor** of **`public MainWindow() { ... }`**:

```
private readonly Library _library = new();

private void New(object sender, RoutedEventArgs e) =>
    _library.New(Display);
```

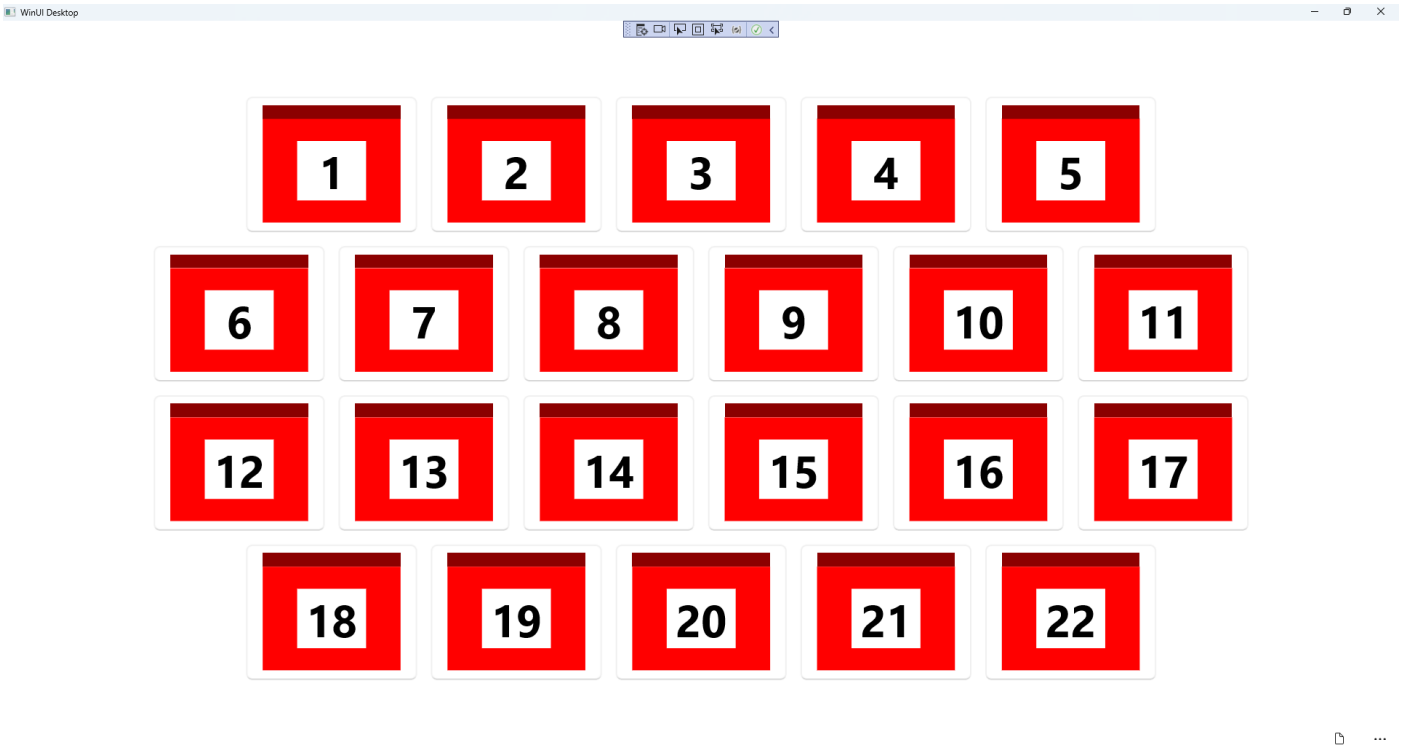
Here an **Instance** of the **Class** of **`Library`** is created then below this is the **Method** of **`New`** that will be used with **Event Handler** from the **XAML**, this **Method** uses Arrow Syntax with the **`=>`** for an Expression Body which is useful when a **Method** only has one line.

Step 17

That completes the **Windows App SDK** Application. In **Visual Studio 2022** from the **Toolbar** select **DealOrNot (Package)** to **Start** the Application.

Step 18

Once running you can then select one of the boxes and an amount will be displayed but each five turns you'll have the chance to take a **Deal** or **Not** and can continue until there is just one box left, which you'll win the amount, but if you took the **Deal** then you win that amount instead, or you can select **New** to start a new game.



Step 19

To **Exit** the **Windows App SDK** Application, select the **Close** button from the top right of the Application as that concludes this **Tutorial** for **Windows App SDK** from tutorialr.com!